Designed for Microsoft® Windows NT® Windows® 98

CD-ROM Included

Microsoft® 程序设计系列

完全修订和更新！

Windows® 程序设计

（第5版）下册

Win32 API 的权威指导

北京大学出版社

Microsoft Press

Windows 程序设计

（第 5 版）

下　册

[美]Charles Petzold　著

北京博彦科技发展有限公司　译

北京大学出版社

·北　京·

第二部分　其他图形知识

第十三章

使用打印机

当使用视频显示器显示文本和图形时，设备无关的概念非常完美，但对于打印机，设备无关的概念又怎样呢？

总的说来，效果也很好。在 Windows 程序中，您可以使用同样的用于视频显示的 GDI 函数，在打印纸上打印文本和图形，在以前讨论的与设备无关的许多问题（多数都与显示表面的尺寸和分辨率，以及颜色容量有关）都可以用相同的方法解决。当然，一台打印机不像阴极射线管那么简单，它们使用的是打印纸。它们之间有一些比较大的差异。例如，我们从来不必考虑视频显示器没有与视频适配器连结好，或者出现“屏幕空间不够”的错误，但打印机脱机和缺纸则是经常会遇到的问题。

我们也不必担心视频适配器不能执行某些图形操作，更不用担心视频适配器能否处理图形，因为如果它不能处理图形，就根本不能使用 Windows。但有些打印机不能打印图形（尽管它们能在 Windows 环境中使用）；绘图仪尽管可以打印矢量图形，却存在与位图有关的实际问题。

以下是其他若干需要考虑的问题：

- 打印机比视频显示器慢。尽管我们没有机会将程序性能调整到最佳状态，却不必担心视频显示器刷新所需的时间。然而，没有人想在做其他工作前一直等打印机完成打印任务。
- 程序可以用新的输出覆盖原有的显示输出，以重新使用视频显示器表面。这对打印机是不可能的，打印机只能走完一整页纸，然后在新的一页纸上打印新的内容。
- 在视频显示器上，不同的应用程序都被窗口化。而对于打印机，不同应用程序的输出必须分成不同的文档或打印作业。

为了给 GDI 的其余部分添加打印机支持功能，Windows 提供了几个只用于打印机

的函数。这些打印机专用的函数(StartDoc、EndDoc、StartPage 和 EndPage)负责将打印机的输出组织到纸张页面上。一个程序调用常用的 GDI 函数在一张纸上"显示"文本和图形,与在屏幕上显示的方式一样。

在第十五、十七和十八章有打印位图、格式化的文本,以及元文件的其他信息。

13.1　打印基础

当您在 Windows 下使用打印机时,实际上启动了一个包含 GDI32 库模块、打印机驱动程序库模块(带.DRV 扩展名)、Windows 后台打印程序,以及起作用的其他一些模块。在为打印机编写程序前,让我们先看一看这个过程是如何进行的。

13.1.1　打印和后台处理

当应用程序要使用打印机时,它首先使用 CreateDC 或 PrintDlg 来获取指向打印机设备描述表的一个句柄,这就使得打印机设备驱动程序库模块被加载到内存(如果还没有加载到内存中的话),并自己进行初始化。然后,程序调用 StartDoc 函数,通知一个新文档开始了。StartDoc 函数是由 GDI 模块来处理的,GDI 模块调用打印机设备驱动程序中的 Control 函数告诉设备驱动程序准备进行打印。

打印一个文档的过程以 StartDoc 调用开始,以 EndDoc 调用结束。这两个调用对于在文档页面上书写文本或者绘制图形的 GDI 命令来说,其作用就像书挡一样。每页本身是这样来划清界限的:调用 StartPage 来开始一页,调用 EndPage 来结束该页。

例如,如果应用程序想在一页纸上画一个椭圆,它首先调用 StartDoc 开始打印任务,然后再调用 StartPage 通知这是新的一页,接着调用 Ellipse,就像在屏幕上画一个椭圆一样。GDI 模块通常将程序对打印机设备描述表做出的 GDI 调用存储在磁盘上的元文件中,该文件名以字符串 ~ EMF(代表"增强型元文件")开始,且以.TMP 为扩展名。然而,打印机驱动程序可能会跳过这一步。

当绘制第一页的 GDI 调用结束时,应用程序调用 EndPage。现在,真正的工作开始了。打印机驱动程序必须把存放在元文件中的各种绘图命令翻译成打印机输出数据。绘制一页图形所需的打印机的输出数据量可能非常大,特别是当打印机没有高级页面制作语言时更是如此。例如,一台每英寸 600 点且使用 8.5 英寸×11 英寸打印纸的激光打印机,如果要定义一个图形页,可能需要 4 兆以上字节的数据。

为此,打印机驱动程序经常使用一种被称作"打印分带"的技术将一页分成若干个被称为"带"的矩形。GDI 模块从打印机驱动程序获取每个带的大小,然后设置一个与当前要处理的带相等的剪裁区,并为从元文件获得的每个绘图函数调用打印机设备驱动程序的 Output 函数,这个过程叫做"将元文件输出到设备驱动程序"。对设备驱动程序定义的页面上的每个带,GDI 模块必须将整个元文件"输出到"设备驱动程序。这个

过程完成以后,就可以删除该元文件。

对每个带,设备驱动程序都会将这些绘图函数转换为在打印机上打印这些图形所需要的输出数据。这种输出数据的格式是特定于打印机的。对点阵打印机,它将是包括图形序列在内的一系列控制序列的集合(打印机驱动程序也能调用在 GDI 模块中的各种"助手"例程,用来协助这种输出的构造)。对于带有高级页面制作语言(如 PostScript)的激光打印机,打印机将用这种语言进行输出。

打印机驱动程序将打印输出的每个带传送到 GDI 模块,随后,GDI 模块将该打印输出存入另一个临时文件中,该临时文件名以字符串 ~SPL 开始,带有.TMP 扩展名。当处理好整个页之后,GDI 模块对后台打印程序进行一个进程间调用,通知它一个新的打印页已经准备好了。然后,应用程序就转向处理下一页。当应用程序处理完所有要打印的页后,它就调用 EndDoc 发出一个信号,表明打印作业已经完成。图 13-1 显示了应用程序、GDI 模块和打印机驱动程序的交互作用过程。

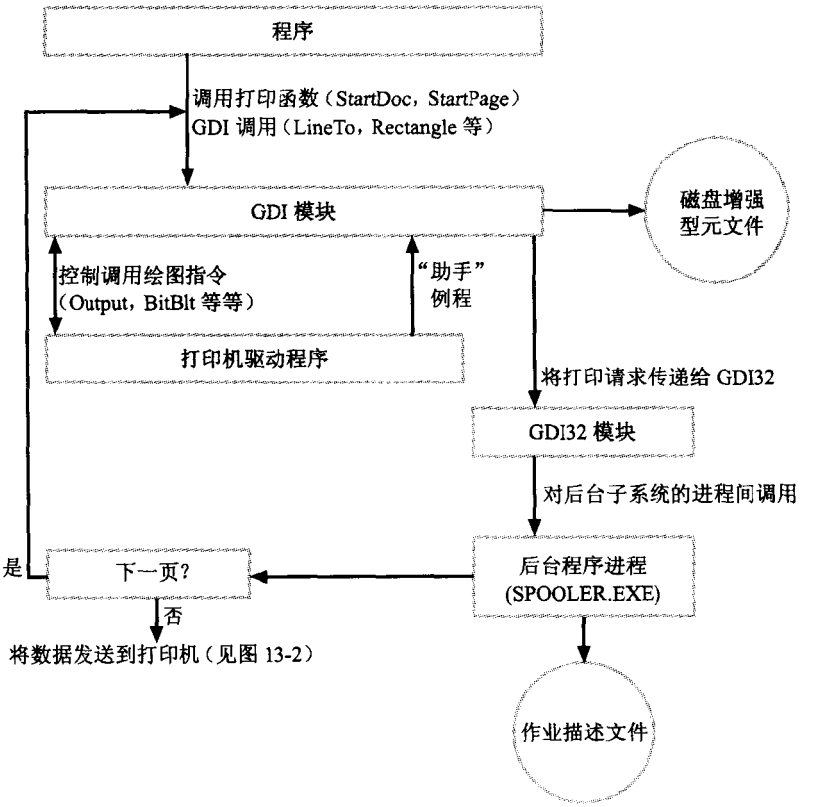


图 13-1 应用程序、GDI 模块、打印机驱动程序和后台程序的交互过程

Windows 后台打印程序实际上是几个部件的一种组合(见表 13-1)。

表 13-1

后台程序部件	说明
打印请求后台程序	将数据流传递给打印提供者
本地打印提供者	为本地打印机创建后台文件
网络打印提供者	为网络打印机创建后台文件
打印处理程序	将后台的设备无关数据转换为特定于目标打印机的格式
端口监视程序	控制连接打印机的端口
语言监视程序	控制可以双向通讯的打印机,设置设备配置并检测打印机状态

后台程序可以减轻应用程序的打印负担。Windows 在启动时就加载后台打印程序,因此,当应用程序开始打印时,它已经是活动的了。当程序打印一个文件时,GDI 模块创建包含打印输出数据的文件。后台打印程序的任务是将这些文件发往打印机。GDI 模块发出一个消息来通知它开始一个新的打印作业,然后它开始读文件,并将文件直接传送到打印机。为了传送这些文件,后台程序为打印机所连接的并口和串口使用各种通讯函数。在后台程序向打印机发送文件的操作结束后,它就删除包含输出数据的临时文件。这个过程如图 13-2 所示。

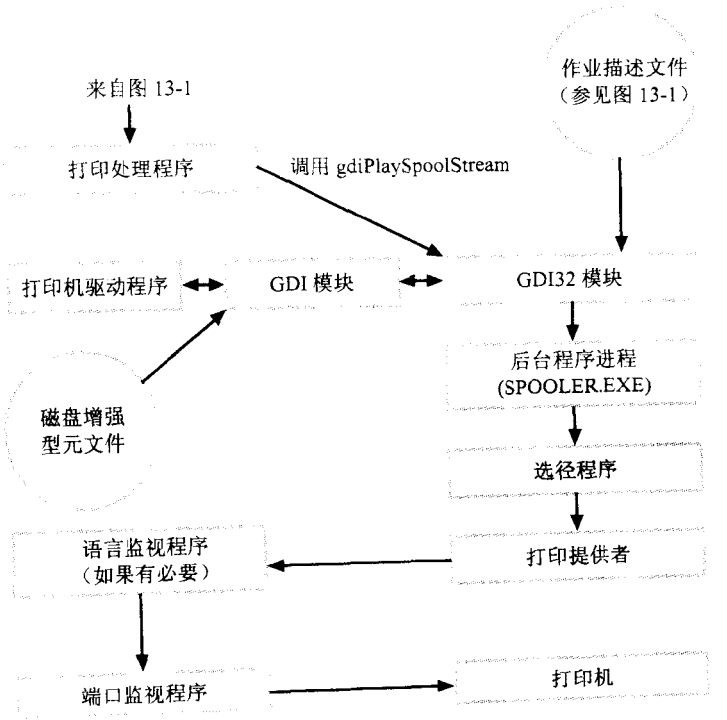


图 13-2　后台打印程序的操作过程

这个过程的大部分对应用程序是透明的。从应用程序角度来看,“打印”只发生在GDI模块将所有打印输出数据保存到磁盘文件中的时候,这以后(如果打印是由第二个线程来操作的,甚至可以在这之前)应用程序可以自由地进行其他操作。真正的文件打印操作成了后台打印程序的任务,而不是应用程序的任务。用户可以暂停打印作业、改变作业的优先级,或取消打印作业。这种管理方式使得应用程序可能比下面这种情况“打印”得更快,即作业以实时方式打印,且必须等到打印完一页后才能处理下一页。

我们已经描述了一般的打印原理,但还有一些例外情况。其中之一是Windows程序要使用打印机时,并非一定需要后台打印程序。用户可以在打印机属性表中关闭打印机的后台操作。

为什么有时用户不希望使用后台操作呢?因为用户可能使用了比Windows后台程序更快的硬件或软件后台打印程序,也可能是打印机在一个自身带有后台程序的网络上使用。一般的规律是,使用一个后台程序比使用两个后台程序更快。去掉Windows后台打印程序可以加快打印速度,因为打印输出数据不必保存在硬盘上,而可以直接输出到打印机,并被外部硬件或软件后台打印程序所接收。

如果没有启用Windows后台程序,GDI模块就不把来自设备驱动程序的打印输出数据存入文件中,而是将这些输出数据直接输出到并行或串行打印端口。与后台程序进行的打印不同,GDI进行的打印潜在地挂起应用程序(特别地,就是指正在进行打印的程序)的操作,直到打印完成。

还有一个例外。通常,GDI模块将定义一页所需的所有函数存入一个增强型元文件中,然后为由驱动程序定义的每个打印分带输出一遍该元文件到打印机驱动程序中。然而,如果打机印驱动程序不需要打印分带的话,就不会创建这个元文件;GDI只需简单地将绘图函数直接送往驱动程序。进一步的变化是,应用程序也可能承担起对打印输出数据进行打印分带的责任,这就使得应用程序中的打印代码更加复杂了,但却免去了GDI模块创建元文件的麻烦。这样,GDI只需简单地为每个带将函数传到打印机驱动程序。

或许您现在已经发现了从一个Windows应用程序进行打印操作要比使用视频显示器的开销大,这样可能出现一些问题——特别是当创建元文件或打印输出文件时,如果GDI模块耗尽了磁盘空间。您可以更多地关心这些问题,并尝试着处理这些问题,也可以置之不理。

应用程序进行打印的第一步是获取打印机设备描述表。

13.1.2 打印机设备描述表

正如在视频显示器上绘图前需要得到设备描述表句柄一样,在打印之前,用户必须获取一个打印机设备描述表句柄。一旦有了这个句柄(并为创建一个新文档调用StartDoc,

以及调用 StartPage 开始一页),就可以用与使用视频显示设备描述表句柄相同的方法来使用打印机设备描述表句柄,该句柄即为各种 GDI 调用的第一个参数。

很多应用程序通过调用 PrintDlg 函数打开一个标准的打印对话框。(本章后面会演示该函数的用法)。这个函数还给用户提供了一个在打印之前更改打印机或者指定其他作业特性的机会。然后,它将打印机设备描述表句柄赋予该应用程序。该函数能够保存应用程序的某些工作。然而,某些应用程序(例如记录本)只获得打印机设备描述表,而不需要对话框。要做到这一点,需要调用 CreateDC 函数。

在第五章,您知道如何通过如下调用来为整个视频显示器获取指向设备描述表的句柄:

```
hdc = CreateDC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
```

您也可以使用该函数来获取打印机设备描述表句柄。然而,对于打印机设备描述表,CreateDC 的一般语法为

```
hdc = CreateDC (NULL, szDeviceName, NULL, pInitializationData) ;
```

pInitializationData 参数一般被置为 NULL,szDeviceName 参数指向一个字符串,以告诉 Windows 打印机设备的名称。在设置设备名称之前,您必须知道有哪些打印机可用。

一个系统可以连接多台打印机,甚至可以有其他程序,如传真软件等,将自己伪装成打印机。不论连接的打印机有多少,都只能有一个被认为是"当前打印机"或者"默认打印机",这是用户最近一次选择的打印机。许多小的 Windows 程序只使用这个打印机来进行打印。

获取默认打印机设备描述表的方式在不断地变化。目前,标准的方法是使用 EnumPrinters 函数来获得。该函数填充一个包含每个所连打印机信息的数组结构。根据所需的细节层次,您还可以选择几种结构之一作为该函数的参数。这些结构的名称为 PRINTER_INFO_x,x 是一个数字。

遗憾的是,使用的结构还要依赖于您的程序是运行在 Windows 98 上,还是运行在 Windows NT 上。程序 13-1 表明 GetPrinterDC 函数在这两种操作系统上都能工作。

程序 13-1　GETPRNDC.C 文件

GETPRNDC.C

```
/*-------------------
   GETPRNDC.C——GetPrinterDC function
   -------------------*/

#include <windows.h>

HDC GetPrinterDC (void)
```

```
{
     DWORD              dwNeeded, dwReturned ;
     HDC                hdc ;
     PRINTER_INFO_4 *   pinfo4 ;
     PRINTER_INFO_5 *   pinfo5 ;

     if (GetVersion () & 0x80000000)          // Windows 98
     {
          EnumPrinters (PRINTER_ENUM_DEFAULT, NULL, 5, NULL,
                        0, &dwNeeded, &dwReturned) ;

          pinfo5 = malloc (dwNeeded) ;

          EnumPrinters (PRINTER_ENUM_DEFAULT, NULL, 5, (PBYTE) pinfo5,
                        dwNeeded, &dwNeeded, &dwReturned) ;

          hdc = CreateDC (NULL, pinfo5 -> pPrinterName, NULL, NULL) ;

          free (pinfo5) ;
     }
     else                                     // Windows NT
     {
          EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 4, NULL,
                        0, &dwNeeded, &dwReturned) ;

          pinfo4 = malloc (dwNeeded) ;

          EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 4, (PBYTE) pinfo4,
                        dwNeeded, &dwNeeded, &dwReturned) ;

          hdc = CreateDC (NULL, pinfo4 -> pPrinterName, NULL, NULL) ;

          free (pinfo4) ;
     }
     return hdc ;
}
```

这些函数使用 GetVersion 函数来确定程序是运行在 Windows 98 上，还是运行在 Windows NT 上。不管是什么操作系统，函数调用 EnumPrinters 两次：一次获取它所需结构的大小；一次填充结构。在 Windows 98 上，函数使用 PRINTER_INFO_5 结构；在 Windows NT 上，函数使用 PRINTER_INFO_4 结构。这些结构在 EnumPrinters 文档（/Platform SDK/Graphics and Multimedia Services/GDI/Printing and Print Spooler/Printing and Print Spooler Reference/Printing and Print Spooler Functions/EnumPrinters, Examples 小节的前面）中说明，它们是“容易而快速”的。

13.1.3 修改后的 DEVCAPS 程序

第五章的 DEVCAPS1 程序只显示了视频显示器的 GetDeviceCaps 函数的一些可用的基本信息。程序 13-2 所示的新版本显示了关于视频显示器和所有连接到系统的打印机的详细信息。

程序 13-2　DEVCAPS2 程序

DEVCAPS2.C

```
/*-------------------------------------
   DEVCAPS2.C——Displays Device Capability Information (Version 2)
                (c) Charles Petzold, 1998
   -------------------------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
void DoBasicInfo        (HDC, HDC, int, int) ;
void DoOtherInfo        (HDC, HDC, int, int) ;
void DoBitCodedCaps     (HDC, HDC, int, int, int) ;

typedef struct
{
int         iMask ;
TCHAR *     szDesc ;
}
BITS ;

#define IDM_DEVMODE       1000

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR   szAppName[] = TEXT ("DevCaps2") ;
     HWND           hwnd ;
     MSG            msg ;
     WNDCLASS       wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
```

```
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, NULL,
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static TCHAR            szDevice[32], szWindowText[64] ;
     static int              cxChar, cyChar, nCurrentDevice = IDM_SCREEN,
                                             nCurrentInfo = IDM_BASIC ;
     static DWORD            dwNeeded, dwReturned ;
     static PRINTER_INFO_4 * pinfo4 ;
     static PRINTER_INFO_5 * pinfo5 ;
     DWORD                   i ;
     HDC                     hdc, hdcInfo ;
     HMENU                   hMenu ;
     HANDLE                  hPrint ;
     PAINTSTRUCT             ps ;
     TEXTMETRIC              tm ;

     switch (message)
     {
     case WM_CREATE :
```

```
            hdc = GetDC (hwnd) ;
            SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
            GetTextMetrics (hdc, &tm) ;
            cxChar = tm.tmAveCharWidth ;
            cyChar = tm.tmHeight + tm.tmExternalLeading ;
            ReleaseDC (hwnd, hdc) ;
                                        // fall through
        case WM_SETTINGCHANGE:
            hMenu = GetSubMenu (GetMenu (hwnd), 0) ;

            while (GetMenuItemCount (hMenu) > 1)
                DeleteMenu (hMenu, 1, MF_BYPOSITION) ;

            // Get a list of all local and remote printers
            //
            // First, find out how large an array we need; this
            // call will fail, leaving the required size in dwNeeded
            //
            // Next, allocate space for the info array and fill it
            //
            // Put the printer names on the menu

            if (GetVersion () & 0x80000000)             // Windows 98
            {
                EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 5, NULL,
                              0, &dwNeeded, &dwReturned) ;

                pinfo5 = malloc (dwNeeded) ;

                EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 5, (PBYTE) pinfo5,
                              dwNeeded, &dwNeeded, &dwReturned) ;

                for (i = 0 ; i < dwReturned ; i++)
                {
                    AppendMenu (hMenu, (i+1) % 16 ? 0 : MF_MENUBARBREAK, i + 1,
                                pinfo5[i].pPrinterName) ;
                }
                free (pinfo5) ;
            }
            else                                        // Windows NT
            {
                EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 4, NULL,
                              0, &dwNeeded, &dwReturned) ;

                pinfo4 = malloc (dwNeeded) ;
                EnumPrinters (PRINTER_ENUM_LOCAL, NULL, 4, (PBYTE) pinfo4,
                              dwNeeded, &dwNeeded, &dwReturned) ;
```

```
            for (i = 0 ; i < dwReturned ; i++)
            {
                AppendMenu (hMenu, (i+1) % 16 ? 0 : MF_MENUBARBREAK, i + 1,
                            pinfo4[i].pPrinterName) ;
            }
            free (pinfo4) ;
        }

        AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
        AppendMenu (hMenu, 0, IDM_DEVMODE, TEXT ("Properties")) ;

        wParam = IDM_SCREEN ;
                                        // fall through
    case WM_COMMAND :
        hMenu = GetMenu (hwnd) ;

        if (LOWORD (wParam) == IDM_SCREEN ||        // IDM_SCREEN & Printers
            LOWORD (wParam) < IDM_DEVMODE)
        {
            CheckMenuItem (hMenu, nCurrentDevice, MF_UNCHECKED) ;
            nCurrentDevice = LOWORD (wParam) ;
            CheckMenuItem (hMenu, nCurrentDevice, MF_CHECKED) ;
        }
        else if (LOWORD (wParam) == IDM_DEVMODE)        // Properties selection
        {
            GetMenuString (hMenu, nCurrentDevice, szDevice,
                           sizeof (szDevice) / sizeof (TCHAR), MF_BYCOMMAND);

            if (OpenPrinter (szDevice, &hPrint, NULL))
            {
                PrinterProperties (hwnd, hPrint) ;
                ClosePrinter (hPrint) ;
            }
        }
        else                                    // info menu items
        {
            CheckMenuItem (hMenu, nCurrentInfo, MF_UNCHECKED) ;
            nCurrentInfo = LOWORD (wParam) ;
            CheckMenuItem (hMenu, nCurrentInfo, MF_CHECKED) ;
        }
        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

    case WM_INITMENUPOPUP :
        if (lParam == 0)
            EnableMenuItem (GetMenu (hwnd), IDM_DEVMODE,
                nCurrentDevice == IDM_SCREEN ? MF_GRAYED : MF_ENABLED) ;
```

```
        return 0 ;

    case WM_PAINT :
        lstrcpy (szWindowText, TEXT ("Device Capabilities: ")) ;

        if (nCurrentDevice == IDM_SCREEN)
        {
            lstrcpy (szDevice, TEXT ("DISPLAY")) ;
            hdcInfo = CreateIC (szDevice, NULL, NULL, NULL) ;
        }
        else
        {
            hMenu = GetMenu (hwnd) ;
            GetMenuString (hMenu, nCurrentDevice, szDevice,
                           sizeof (szDevice), MF_BYCOMMAND) ;
            hdcInfo = CreateIC (NULL, szDevice, NULL, NULL) ;
        }

        lstrcat (szWindowText, szDevice) ;
        SetWindowText (hwnd, szWindowText) ;

        hdc = BeginPaint (hwnd, &ps) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;

        if (hdcInfo)
        {
            switch (nCurrentInfo)
            {
            case IDM_BASIC :
                DoBasicInfo (hdc, hdcInfo, cxChar, cyChar) ;
                break ;

            case IDM_OTHER :
                DoOtherInfo (hdc, hdcInfo, cxChar, cyChar) ;
                break ;

            case IDM_CURVE :
            case IDM_LINE :
            case IDM_POLY :
            case IDM_TEXT :
                DoBitCodedCaps (hdc, hdcInfo, cxChar, cyChar,
                                nCurrentInfo - IDM_CURVE) ;
                break ;
            }
            DeleteDC (hdcInfo) ;
        }
```

```
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

void DoBasicInfo (HDC hdc, HDC hdcInfo, int cxChar, int cyChar)
{
        static struct
        {
            int         nIndex ;
            TCHAR * szDesc ;
        }
        info[] =
        {
            HORZSIZE,       TEXT ("HORZSIZE       Width in millimeters:"),
            VERTSIZE,       TEXT ("VERTSIZE       Height in millimeters:"),
            HORZRES,        TEXT ("HORZRES        Width in pixels:"),
            VERTRES,        TEXT ("VERTRES        Height in raster lines:"),
            BITSPIXEL,      TEXT ("BITSPIXEL      Color bits per pixel:"),
            PLANES,         TEXT ("PLANES         Number of color planes:"),
            NUMBRUSHES,     TEXT ("NUMBRUSHES     Number of device brushes:"),
            NUMPENS,        TEXT ("NUMPENS        Number of device pens:"),
            NUMMARKERS,     TEXT ("NUMMARKERS     Number of device markers:"),
            NUMFONTS,       TEXT ("NUMFONTS       Number of device fonts:"),
            NUMCOLORS,      TEXT ("NUMCOLORS      Number of device colors:"),
            PDEVICESIZE,    TEXT ("PDEVICESIZE    Size of device structure:"),
            ASPECTX,        TEXT ("ASPECTX        Relative width of pixel:"),
            ASPECTY,        TEXT ("ASPECTY        Relative height of pixel:"),
            ASPECTXY,       TEXT ("ASPECTXY       Relative diagonal of pixel:"),
            LOGPIXELSX,     TEXT ("LOGPIXELSX     Horizontal dots per inch:"),
            LOGPIXELSY,     TEXT ("LOGPIXELSY     Vertical dots per inch:"),
            SIZEPALETTE,    TEXT ("SIZEPALETTE    Number of palette entries:"),
            NUMRESERVED,    TEXT ("NUMRESERVED    Reserved palette entries:"),
            COLORRES,       TEXT ("COLORRES       Actual color resolution:"),
            PHYSICALWIDTH,  TEXT ("PHYSICALWIDTH  Printer page pixel width:"),
            PHYSICALHEIGHT, TEXT ("PHYSICALHEIGHT Printer page pixel height:"),
            PHYSICALOFFSETX,TEXT ("PHYSICALOFFSETX Printer page x offset:"),
            PHYSICALOFFSETY,TEXT ("PHYSICALOFFSETY Printer page y offset:")
        } ;
        int     i ;
        TCHAR szBuffer[80] ;

        for (i = 0 ; i < sizeof (info) / sizeof (info[0]) ; i++)
```

```
        TextOut (hdc, cxChar, (i + 1) * cyChar, szBuffer,
            wsprintf (szBuffer, TEXT ("%-45s%8d"), info[i].szDesc,
                GetDeviceCaps (hdcInfo, info[i].nIndex))) ;
}

void DoOtherInfo (HDC hdc, HDC hdcInfo, int cxChar, int cyChar)
{
    static BITS clip[] =
    {
        CP_RECTANGLE,       TEXT ("CP_RECTANGLE         Can Clip To Rectangle:")
    } ;

    static BITS raster[] =
    {
        RC_BITBLT,          TEXT ("RC_BITBLT            Capable of simple BitBlt:"),
        RC_BANDING,         TEXT ("RC_BANDING           Requires banding support:"),
        RC_SCALING,         TEXT ("RC_SCALING           Requires scaling support:"),
        RC_BITMAP64,        TEXT ("RC_BITMAP64          Supports bitmaps > 64K:"),
        RC_GDI20_OUTPUT,    TEXT ("RC_GDI20_OUTPUT      Has 2.0 output calls:"),
        RC_DI_BITMAP,       TEXT ("RC_DI_BITMAP         Supports DIB to memory:"),
        RC_PALETTE,         TEXT ("RC_PALETTE           Supports a palette:"),
        RC_DIBTODEV,        TEXT ("RC_DIBTODEV          Supports bitmap conversion:"),
        RC_BIGFONT,         TEXT ("RC_BIGFONT           Supports fonts > 64K:"),
        RC_STRETCHBLT,      TEXT ("RC_STRETCHBLT        Supports StretchBlt:"),
        RC_FLOODFILL,       TEXT ("RC_FLOODFILL         Supports FloodFill:"),
        RC_STRETCHDIB,      TEXT ("RC_STRETCHDIB        Supports StretchDIBits:")
    } ;

    static TCHAR * szTech[] = { TEXT ("DT_PLOTTER (Vector plotter)"),
                                TEXT ("DT_RASDISPLAY (Raster display)"),
                                TEXT ("DT_RASPRINTER (Raster printer)"),
                                TEXT ("DT_RASCAMERA (Raster camera)"),
                                TEXT ("DT_CHARSTREAM (Character stream)"),
                                TEXT ("DT_METAFILE (Metafile)"),
                                TEXT ("DT_DISPFILE (Display file)") } ;
    int             i ;
    TCHAR           szBuffer[80] ;

    TextOut (hdc, cxChar, cyChar, szBuffer,
        wsprintf (szBuffer, TEXT ("%-24s%04XH"), TEXT ("DRIVERVERSION:"),
            GetDeviceCaps (hdcInfo, DRIVERVERSION))) ;

    TextOut (hdc, cxChar, 2 * cyChar, szBuffer,
        wsprintf (szBuffer, TEXT ("%-24s%-40s"), TEXT ("TECHNOLOGY:"),
            szTech[GetDeviceCaps (hdcInfo, TECHNOLOGY)])) ;
```

```
     TextOut (hdc, cxChar, 4 * cyChar, szBuffer,
          wsprintf (szBuffer, TEXT ("CLIPCAPS (Clipping capabilities)"))) ;

     for (i = 0 ; i < sizeof (clip) / sizeof (clip[0]) ; i++)
          TextOut (hdc, 9 * cxChar, (i + 6) * cyChar, szBuffer,
               wsprintf (szBuffer, TEXT ("%-45s %3s"), clip[i].szDesc,
                    GetDeviceCaps (hdcInfo, CLIPCAPS) & clip[i].iMask ?
                         TEXT ("Yes") : TEXT ("No"))) ;

     TextOut (hdc, cxChar, 8 * cyChar, szBuffer,
          wsprintf (szBuffer, TEXT ("RASTERCAPS (Raster capabilities)"))) ;

     for (i = 0 ; i < sizeof (raster) / sizeof (raster[0]) ; i++)
          TextOut (hdc, 9 * cxChar, (i + 10) * cyChar, szBuffer,
               wsprintf (szBuffer, TEXT ("%-45s %3s"), raster[i].szDesc,
                    GetDeviceCaps (hdcInfo, RASTERCAPS) & raster[i].iMask ?
                         TEXT ("Yes") : TEXT ("No"))) ;
}

void DoBitCodedCaps (HDC hdc, HDC hdcInfo, int cxChar, int cyChar, int iType)
{
     static BITS curves[] =
     {
          CC_CIRCLES,          TEXT ("CC_CIRCLES          Can do circles:"),
          CC_PIE,              TEXT ("CC_PIE              Can do pie wedges:"),
          CC_CHORD,            TEXT ("CC_CHORD            Can do chord arcs:"),
          CC_ELLIPSES,         TEXT ("CC_ELLIPSES         Can do ellipses:"),
          CC_WIDE,             TEXT ("CC_WIDE             Can do wide borders:"),
          CC_STYLED,           TEXT ("CC_STYLED           Can do styled borders:"),
          CC_WIDESTYLED,       TEXT ("CC_WIDESTYLED       Can do wide and styled borders:"),
          CC_INTERIORS,        TEXT ("CC_INTERIORS        Can do interiors:")
     } ;

     static BITS lines[] =
     {
          LC_POLYLINE,         TEXT ("LC_POLYLINE         Can do polyline:"),
          LC_MARKER,           TEXT ("LC_MARKER           Can do markers:"),
          LC_POLYMARKER,       TEXT ("LC_POLYMARKER       Can do polymarkers"),
          LC_WIDE,             TEXT ("LC_WIDE             Can do wide lines:"),
          LC_STYLED,           TEXT ("LC_STYLED           Can do styled lines:"),
          LC_WIDESTYLED,       TEXT ("LC_WIDESTYLED       Can do wide and styled lines:"),
          LC_INTERIORS,        TEXT ("LC_INTERIORS        Can do interiors:")
     } ;

     static BITS poly[] =
     {
          PC_POLYGON,
```

```
        TEXT ("PC_POLYGON      Can do alternate fill polygon:"),
    PC_RECTANGLE,     TEXT ("PC_RECTANGLE      Can do rectangle:"),
    PC_WINDPOLYGON,
        TEXT ("PC_WINDPOLYGON Can do winding number fill polygon:"),
    PC_SCANLINE,      TEXT ("PC_SCANLINE       Can do scanlines:"),
    PC_WIDE,          TEXT ("PC_WIDE           Can do wide borders:"),
    PC_STYLED,        TEXT ("PC_STYLED         Can do styled borders:"),
    PC_WIDESTYLED,
        TEXT ("PC_WIDESTYLED Can do wide and styled borders:"),
    PC_INTERIORS,     TEXT ("PC_INTERIORS      Can do interiors:")
};

static BITS text[] =
{
    TC_OP_CHARACTER,
        TEXT ("TC_OP_CHARACTER Can do character output precision:"),
    TC_OP_STROKE,
        TEXT ("TC_OP_STROKE    Can do stroke output precision:"),
    TC_CP_STROKE,
        TEXT ("TC_CP_STROKE    Can do stroke clip precision:"),
    TC_CR_90,
        TEXT ("TC_CP_90        Can do 90 degree character rotation:"),
    TC_CR_ANY,
        TEXT ("TC_CR_ANY       Can do any character rotation:"),
    TC_SF_X_YINDEP,
        TEXT ("TC_SF_X_YINDEP  Can do scaling independent of X and Y:"),
    TC_SA_DOUBLE,
        TEXT ("TC_SA_DOUBLE    Can do doubled character for scaling:"),
    TC_SA_INTEGER,
        TEXT ("TC_SA_INTEGER   Can do integer multiples for scaling:"),
    TC_SA_CONTIN,
        TEXT ("TC_SA_CONTIN    Can do any multiples for exact scaling:"),
    TC_EA_DOUBLE,
        TEXT ("TC_EA_DOUBLE    Can do double weight characters:"),
    TC_IA_ABLE,       TEXT ("TC_IA_ABLE        Can do italicizing:"),
    TC_UA_ABLE,       TEXT ("TC_UA_ABLE        Can do underlining:"),
    TC_SO_ABLE,       TEXT ("TC_SO_ABLE        Can do strikeouts:"),
    TC_RA_ABLE,       TEXT ("TC_RA_ABLE        Can do raster fonts:"),
    TC_VA_ABLE,       TEXT ("TC_VA_ABLE        Can do vector fonts:")
};

static struct
{
    int       iIndex ;
```

```
        TCHAR * szTitle ;
        BITS    ( * pbits)[] ;
        int     iSize ;
    }
    bitinfo[] =
    {
        CURVECAPS, TEXT ("CURVCAPS (Curve Capabilities)"),
                (BITS ( * )[]) curves, sizeof (curves) / sizeof (curves[0]),
        LINECAPS, TEXT ("LINECAPS (Line Capabilities)"),
                (BITS ( * )[]) lines, sizeof (lines) / sizeof (lines[0]),
        POLYGONALCAPS, TEXT ("POLYGONALCAPS (Polygonal Capabilities)"),
                (BITS ( * )[]) poly, sizeof (poly) / sizeof (poly[0]),
        TEXTCAPS, TEXT ("TEXTCAPS (Text Capabilities)"),
                (BITS ( * )[]) text, sizeof (text) / sizeof (text[0])
    } ;

    static TCHAR szBuffer[80] ;
    BITS         ( * pbits)[] = bitinfo[iType].pbits ;
    int          i, iDevCaps = GetDeviceCaps (hdcInfo, bitinfo[iType].iIndex) ;

    TextOut (hdc, cxChar, cyChar, bitinfo[iType].szTitle,
             lstrlen (bitinfo[iType].szTitle)) ;

    for (i = 0 ; i < bitinfo[iType].iSize ; i++)
        TextOut (hdc, cxChar, (i + 3) * cyChar, szBuffer,
            wsprintf (szBuffer, TEXT ("%-55s %3s"), ( * pbits)[i].szDesc,
                iDevCaps & ( * pbits)[i].iMask ? TEXT ("Yes") : TEXT ("No")));
}
```

DEVCAPS2.RC(摘录)

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
DEVCAPS2 MENU DISCARDABLE
BEGIN
    POPUP "&Device"
    BEGIN
        MENUITEM "&Screen",                     IDM_SCREEN, CHECKED
    END
    POPUP "&Capabilities"
    BEGIN
        MENUITEM "&Basic Information",          IDM_BASIC
        MENUITEM "&Other Information",          IDM_OTHER
        MENUITEM "&Curve Capabilities",         IDM_CURVE
```

```
        MENUITEM "&Line Capabilities",          IDM_LINE
        MENUITEM "&Polygonal Capabilities",     IDM_POLY
        MENUITEM "&Text Capabilities",          IDM_TEXT
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by DevCaps2.rc

#define IDM_SCREEN          40001
#define IDM_BASIC           40002
#define IDM_OTHER           40003
#define IDM_CURVE           40004
#define IDM_LINE            40005
#define IDM_POLY            40006
#define IDM_TEXT            40007
```

因为 DEVCAPS2 只获取打印机的信息描述表,即使打印机使用的输出端口为"none",用户仍然可以从 DEVCAPS2 的菜单中选择所需的打印机。如果用户想比较不同打印机的功能,可以先用打印机文件夹添加各种打印机驱动程序。

13.1.4 PrinterProperties 调用

DEVCAPS2 程序的 Device 菜单上还有一个 Properties 选项。要使用这个选项,首先从 Device 菜单中选择一个打印机,然后再选择 Properties,这时弹出一个对话框。对话框从何而来呢? 它是打印机驱动程序提供的,最起码提供给用户选择纸张尺寸的选项。大多数打印机驱动程序也可以让用户在"纵向(portrait)"或"横向(landscape)"模式中进行选择。在纵向模式(一般为默认模式)下,纸的短边是顶部;在横向模式下,纸的长边是顶部。如果更改模式,则所做的更改将在 DEVCAPS2 程序从 GetDeviceCaps 函数获取的信息中反映出来:水平尺寸和分辨率将与垂直尺寸和分辨率交换。彩色绘图仪的 Properties 对话框将有更多内容,它们要求用户设置安装在绘图仪上的笔的颜色,以及使用的绘图纸(或透明胶片)的型号。

所有打印机驱动程序都包含一个 ExtDeviceMode 输出函数,它调用对话框,并保存用户输入的信息。有些打印机驱动程序也将这些信息保存在注册表的自己的那个部分中,有些则不然。下次 Windows 使用该打印机时将读取这些存储的信息。

允许用户选择打印机的 Windows 程序通常只要调用 PrintDlg 即可(本章后面将演示用法)。此有用的函数在准备打印时负责和用户之间的所有通讯工作,并负责处理用户要求的所有更改。当用户单击 Properties 按钮时,PrintDlg 还会激活属性表格对话框。

程序还可以通过直接调用打印机驱动程序的 ExtDeviceMode 或 ExtDeveModeProp-

Sheet 函数来显示打印机的属性对话框，然而，我不鼓励您这样做。像 DEVCAPS2 那样，通过调用 PrinterProperties 来激活对话框要好得多。

PrinterProperties 请求打印机对象的句柄，您可以通过 OpenPrinter 函数来得到。当用户取消属性表格对话框时，PrinterProperties 返回，然后通过调用 ClosePrinter 来释放打印机句柄。DEVCAPS2 是这样做到的。

程序首先取得当前在 Device 菜单中选择的打印机名称，并将其存入一个名为 szDevice 的字符数组中：

```
GetMenuString (hMenu, nCurrentDevice, szDevice,
               sizeof (szDevice) / sizeof (TCHAR), MF_BYCOMMAND) ;
```

然后，使用 OpenPrinter 获得该设备的句柄。如果调用成功，那么程序接着调用 PrinterProperties，激活对话框，然后调用 ClosePrinter，释放设备句柄：

```
if (OpenPrinter (szDevice, &hPrint, NULL))
{
     PrinterProperties (hwnd, hPrint) ;
     ClosePrinter (hPrint) ;
}
```

13.1.5 检查 BitBlt 能力

您可以用 GetDeviceCaps 函数来获取页中打印区的尺寸和分辨率(通常，该区域不能与整张纸的大小相同)。如果用户想自己进行缩放操作，也可以获得相对的像素宽度和高度。

关于打印机各种能力的大多数信息是用于 GDI 而不是用于应用程序的。通常，在打印机不能做某件事时，GDI 会模拟出。然而，有一种能力应用程序应该检查。

以 RASTERCAPS("光栅支持")参数调用 GetDeviceCaps，它返回的 RC_BITBLT 位包含了一个重要的打印机特性，该位指示设备是否能进行位块传送。大多数点阵打印机、激光打印机和喷墨打印机都能进行位块传送，而大多数绘图仪却不能。不能处理位块传送的设备不支持下列 GDI 函数：CreateCompatibleDC、CreateCompatibleBitmap、PatBlt、BitBlt、StretchBlt、GrayString、DrawIcon、SetPixel、GetPixel、FloodFill、ExtFloodFill、FillRgn、FrameRgn、InvertRgn、PaintRgn、FillRect、FrameRect，以及 InvertRect。这是在视频显示器上使用 GDI 函数与在打印机上使用它们的唯一重要区别。

13.1.6 最简单的打印程序

现在可以开始打印了，我们尽可能简单地开始。事实上，我们的第一个程序只是让打印机走纸换页而已。程序 13-3 的 FORMFEED 程序，演示了打印所需的最小需求。

程序 13-3　FORMFEED 程序

FORMFEED.C

```
/*----------------------
   FORMFEED.C——Advances printer to next page
                (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>

HDC GetPrinterDC (void) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpszCmdLine, int iCmdShow)
{
     static DOCINFO  di = { sizeof (DOCINFO), TEXT ("FormFeed") } ;
     HDC             hdcPrint = GetPrinterDC () ;

     if (hdcPrint != NULL)
     {
          if (StartDoc (hdcPrint, &di) > 0)
               if (StartPage (hdcPrint) > 0 && EndPage (hdcPrint) > 0)
                    EndDoc (hdcPrint) ;

          DeleteDC (hdcPrint) ;
     }
     return 0 ;
}
```

这个程序也需要前面程序 13-1 中的 GETPRNDC.C 文件。

除了获取打印机设备描述表(然后再删除它)外,程序只调用了我们在本章前面讨论过的 4 个打印函数。FORMFEED 首先调用 StartDoc 开始一个新的文档,它测试从 StartDoc 返回的值,只有返回值是正数时,才继续下去:

```
if (StartDoc (hdcPrint, &di) > 0)
```

StartDoc 的第二个参数是指向 DOCINFO 结构的指针。该结构在第一个域包含了结构的大小,在第二个域包含了文本串 FormFeed。当文件正在被打印或者在等待打印时,这个串将出现在打印机任务队列中的 Document Name 列中。通常,该串包含进行打印的应用程序名和被打印的文件名。

如果 StartDoc 成功(由一个正的返回值表示),那么 FORMFEED 调用 StartPage,紧接着立即调用 EndPage。这段代码让打印机走纸,换到新的一页,并再次对返回值进行测试:

```
if (StartPage (hdcPrint) > 0 && EndPage (hdcPrint) > 0)
```

最后，如果不出错，就结束文档：

```
EndDoc (hdcPrint) ;
```

要注意的是，只有当不出错时，才调用 EndDoc 函数。如果其他打印函数中的某一个返回错误代码，那么 GDI 实际上已经中断了文档的打印。如果打印机当前未打印，这种错误代码通常会使打印机复位。测试打印函数的返回值是检测错误的最简单方法。如果您想向用户报告错误，就必须调用 GetLastError 来确定错误。

如果您写过 MS-DOS 下的简单走纸程序，就应该知道，对于大多数打印机，ASCII 码 12(Ctrl-L)启动走纸。为什么不简单地使用 C 的库函数 open，然后用 write 输出 ASCII 码 12 呢？当然，您完全可以这么做，但是必须确定打印机连接的是串行口还是并行口。然后还要确定另外的程序(例如，打印后台程序)是不是正在使用打印机。您并不希望在文档的中间出现走纸，对不对？最后，您还必须确定 ASCII 码 12 是不是连接打印机的走纸符，因为并非所有打印机的走纸符都是 12。事实上，在 PostScript 中的走纸命令就不是 12，而是单词 showpage。

一句话，不要试图绕过 Windows，而应该坚持使用 Windows 函数进行打印。

13.2　打印图形和文本

在一个 Windows 程序中，打印所需的额外开销通常比 FORMFEED 程序多得多，而且还要用 GDI 函数来实际打印一些东西。我们现在来写一个打印一页文本和图形的程序，采用 FORMFEED 程序中的方法，并加入一些新东西。该程序将有三个版本 PRINT1、PRINT2 和 PRINT3，为避免代码重复，每个程序都用前面所示的 GETPRNDC.C 文件和 PRINT.C 文件中的函数，如程序 13-4 所示。

程序 13-4　在 PRINT1，PRINT2 和 PRINT3 程序中使用的 PRINT.C 文件

PRINT.C

```
/*-------------------------------
   PRINT.C——Common routines for Print1, Print2, and Print3
  -------------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
BOOL PrintMyPage (HWND) ;

extern HINSTANCE  hInst ;
extern TCHAR      szAppName[] ;
extern TCHAR      szCaption[] ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hInst = hInstance ;

     hwnd = CreateWindow (szAppName, szCaption,
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

void PageGDICalls (HDC hdcPrn, int cxPage, int cyPage)
{
     static TCHAR szTextStr[] = TEXT ("Hello, Printer!") ;
```

```
        Rectangle (hdcPrn, 0, 0, cxPage, cyPage) ;

        MoveToEx (hdcPrn, 0, 0, NULL) ;
        LineTo   (hdcPrn, cxPage, cyPage) ;
        MoveToEx (hdcPrn, cxPage, 0, NULL) ;
        LineTo   (hdcPrn, 0, cyPage) ;

        SaveDC (hdcPrn) ;

        SetMapMode          (hdcPrn, MM_ISOTROPIC) ;
        SetWindowExtEx      (hdcPrn, 1000, 1000, NULL) ;
        SetViewportExtEx    (hdcPrn, cxPage / 2, -cyPage / 2, NULL) ;
        SetViewportOrgEx    (hdcPrn, cxPage / 2,  cyPage / 2, NULL) ;

        Ellipse (hdcPrn, -500, 500, 500, -500) ;

        SetTextAlign (hdcPrn, TA_BASELINE | TA_CENTER) ;
        TextOut (hdcPrn, 0, 0, szTextStr, lstrlen (szTextStr)) ;

        RestoreDC (hdcPrn, -1) ;
    }

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static int        cxClient, cyClient ;
    HDC               hdc ;
    HMENU             hMenu ;
    PAINTSTRUCT       ps ;

    switch (message)
    {
    case WM_CREATE:
        hMenu = GetSystemMenu (hwnd, FALSE) ;
        AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
        AppendMenu (hMenu, 0, 1, TEXT ("&Print")) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_SYSCOMMAND:
        if (wParam == 1)
        {
            if (!PrintMyPage (hwnd))
                MessageBox (hwnd, TEXT ("Could not print page!"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
```

```
            return 0 ;
        }
        break ;

    case WM_PAINT :
        hdc = BeginPaint (hwnd, &ps) ;

        PageGDICalls (hdc, cxClient, cyClient) ;

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY :
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

PRINT.C 包括函数 WinMain、WndProc，以及一个 PageGDICalls 函数。PageGDICalls 函数接收打印机设备描述表句柄和两个包含打印页面宽度及高度的变量。这个函数还负责画一个包围整个页面的矩形，有两条对角线，页中间有一个椭圆（其直径是打印机高度和宽度中较小的那个的一半），文本“Hello, Printer!”位于椭圆中间。

处理 WM_CREATE 消息时，WndProc 将一个 Print 选项添加到系统菜单上。选择该选项将调用 PrintMyPage，此函数的功能在程序的三个版本中将不断增强。当打印成功时，PrintMyPage 返回 TRUE 值，如果遇到错误，则返回 FALSE。如果 PrintMyPage 返回 FALSE，WndProc 就会显示一个消息框以告知用户发生了错误。

13.2.1　打印的基本过程

打印程序的第一个版本是 PRINT1，见程序 13-5。经编译后即可运行此程序，然后从系统菜单中选择 Print。接着，GDI 将必要的打印机输出保存在一个临时文件中，然后后台程序将它发送给打印机。

程序 13-5　PRINT1 程序

PRINT1.C

```
/*----------------------
   PRINT1.C——Bare Bones Printing
             (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>
```

```
HDC GetPrinterDC (void) ;                    // in GETPRNDC.C
void PageGDICalls (HDC, int, int) ;          // in PRINT.C

HINSTANCE  hInst ;
TCHAR      szAppName[ ] = TEXT ("Print1") ;
TCHAR      szCaption[ ] = TEXT ("Print Program 1") ;

BOOL PrintMyPage (HWND hwnd)
{
     static DOCINFO  di = { sizeof (DOCINFO), TEXT ("Print1: Printing") } ;
     BOOL            bSuccess = TRUE ;
     HDC             hdcPrn ;
     int             xPage, yPage ;

     if (NULL == (hdcPrn = GetPrinterDC ()))
          return FALSE ;
     xPage = GetDeviceCaps (hdcPrn, HORZRES) ;
     yPage = GetDeviceCaps (hdcPrn, VERTRES) ;

     if (StartDoc (hdcPrn, &di) > 0)
     {
          if (StartPage (hdcPrn) > 0)
          {
               PageGDICalls (hdcPrn, xPage, yPage) ;

               if (EndPage (hdcPrn) > 0)
                    EndDoc (hdcPrn) ;
               else
                    bSuccess = FALSE ;
          }
     }
     else
          bSuccess = FALSE ;

     DeleteDC (hdcPrn) ;
     return bSuccess ;
}
```

我们来看看 PRINT1.C 中的代码。如果 PrintMyPage 不能获取打印机的设备描述表句柄,它就返回 FALSE,并且 WndProc 显示消息框指出错误。如果函数成功获取了设备描述表句柄,它就通过调用 GetDeviceCaps 来确定页面的水平和垂直大小(以像素为单位):

```
xPage = GetDeviceCaps (hdcPrn, HORZRES) ;
yPage = GetDeviceCaps (hdcPrn, VERTRES) ;
```

这不是纸的全部大小,只是纸的可打印区域。调用后,除了 PRINT1 在 StartPage 和 End-

Page调用之间调用 PageGDICalls，PRINT1 的 PrintMyPage 函数中的代码在结构上与 FORMFEED中的代码相同。仅当调用 StartDoc、StartPage 和 EndPage 都成功时，PRINT1 才调用 EndDoc 打印函数。

13.2.2 使用中止过程来取消打印

对于大的文档，程序应该给用户提供在应用程序打印期间取消打印任务的方便方式。也许用户只要打印文档中的一页，而不是打印全部的 537 页。所以应该在打印完全部 537 页之前纠正这个错误。

在一个程序内取消一个打印任务要求的一种被称为“中止过程”的技术。中止过程在程序中只是一个较小的输出函数，用户可以使用 SetAbortProc 函数将该函数的地址传给 Windows。然后 GDI 在打印时，重复调用该过程，并不断地问询：“我是否应该继续打印?”

我们看看将中止过程添加到打印逻辑中需要些什么，然后检查一些分支。中止过程一般命名为 AbortProc，其形式为：

```
BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
    [ other program lines ]
}
```

打印前，您必须通过调用 SetAbortProc 来登记中止过程：

```
SetAbortProc (hdcPrn, AbortProc) ;
```

在调用 StartDoc 之前调用上面的函数，打印完成后没有必要清除中止过程。

在处理 EndPage 调用时(即在将元文件放入设备驱动程序并创建临时打印文件时)，GDI 常常调用中止过程。参数 hdcPrn 是打印机设备描述表句柄。如果一切正常，iCode 参数是 0，如果 GDI 模块在生成临时文件时耗尽了磁盘空间，iCode 就是 SP_OUTOFDISK。

如果要打印作业继续打印，那么 AbortProc 必须返回 TRUE(非 0)；如果中止打印作业，就返回 FALSE(0)。中止过程可以被简化为如下所示的形式：

```
BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
     MSG   msg ;

     while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
```

```
    return TRUE ;
}
```

这个函数看起来有点特殊,其实它看起来像是消息循环。用户会注意到,这个"消息循环"调用 PeekMessage 而不是 GetMessage。我在第五章的 RANDRECT 程序中已讨论过 PeekMessage。还可以记起,PeekMessage 将控制返回给程序,而不管程序的消息队列中是否有消息存在。

只要 PeekMessage 返回 TRUE,那么 AbortProc 函数中的消息循环就重复调用 PeekMessage。TRUE 值表示 PeekMessage 已经检索到了一个消息,该消息可以通过 TranslateMessage 和 DispatchMessage 发送到程序的窗口过程。若程序的消息队列中没有消息,则 PeekMessage 的返回值为 FALSE,因此,AbortProc 将控制返回给 Windows。

13.2.3 Windows 如何使用 AbortProc

当程序进行打印时,大部分工作发生在调用 EndPage 的过程中。调用 EndPage 前,程序每调用一次 GDI 绘图函数,GDI 模块只是简单地将另一个记录加到磁盘上的元文件中。当 GDI 得到 EndPage 后,对打印页中由设备驱动程序定义的每个带,GDI 都将该元文件送入设备驱动程序中。然后,GDI 将打印机驱动程序创建的打印输出保存到一个文件中。如果没有启用后台打印,那么 GDI 模块必须自己将该打印输出写入打印机。

在 EndPage 调用期间,GDI 模块调用您设置的中止过程。通常 iCode 参数为 0,但如果由于存在未打印的其他临时文件而引起 GDI 运行时磁盘空间不够,那么 iCode 参数就为 SP_OUTOFDISK(通常不检查这个值,但是如果愿意,您可以进行检查)。中止过程随后进入 PeekMessage 循环,从自己的消息队列中检索消息。

如果在程序的消息队列中没有消息,PeekMessage 返回 FALSE,然后中止过程跳出它的消息循环,并给 GDI 模块返回一个 TRUE 值,指示打印应该继续进行。然后 GDI 模块继续处理 EndPage 调用。

如果有错误发生,那么 GDI 将中止打印过程,这样,中止过程的主要目的是允许用户取消打印。为此,我们还需要一个显示 Cancel 按钮的对话框,让我们采用两个独立的步骤。首先,我们在创建 PRINT2 程序时添加一个中止过程,然后在 PRINT3 中添加一个带有"Cancel"按钮的对话框,使中止过程可用。

13.2.4 实现中止过程

现在快速复习一下中止过程的机制。可以定义一个如下所示的中止过程:

```
BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
    MSG    msg ;
```

```
    while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return TRUE ;
}
```

当您想打印什么时，使用下面的调用将指向中止过程的指针传给 Windows：

```
SetAbortProc (hdcPrn, AbortProc) ;
```

在调用 StartDoc 之前进行这个调用就行了。

不过，事情没这么简单。我们忽视了 AbortProc 过程中 PeekMessage 循环这个问题，它是个很大的问题。仅当程序处于打印过程时，才调用 AbortProc 过程。如果在 AbortProc 中检索到一个消息，并把它传送给窗口过程，那么就会发生某些非常令人讨厌的事情：用户可以从菜单中再次选择 Print，但程序已经处于打印例程之中。当程序在打印前一个文件的同时，用户也可以把一个新文件加载到程序里。用户甚至可以退出程序！如果这种情况发生了，所有用户程序的窗口都将被清除。最终将从打印例程中返回，但是除了退到不再有效的窗口例程外，您无处可去。

这种东西会把人搞得晕头转向，而我们的程序对此并无准备。正是由于这个原因，当设置放弃过程时，首先应禁用程序窗口，使它不能接受键盘和鼠标输入。可以用以下函数完成这项工作：

```
EnableWindow (hwnd, FALSE) ;
```

它可以禁止键盘和鼠标的输入进入消息队列。因此在打印过程中，用户不能对程序做任何工作。当打印完成时，应重新允许窗口接受输入：

```
EnableWindow (hwnd, TRUE) ;
```

您可能要问，既然没有键盘或鼠标消息进入消息队列，为什么我们还要进行 AbortProc 中的 TranslateMessage 和 DispatchMessage 调用呢？实际上并不一定非得需要 TranslateMessage，但是，我们必须使用 DispatchMessage，处理 WM_PAINT 消息进入消息队列中的情况。如果 WM_PAINT 消息没有得到窗口过程中的 BeginPaint 和 EndPaint 的适当处理，由于 PeekMessage 不再返回 FALSE，该消息就会滞留在队列中，并会妨碍工作。

当在打印期间禁用窗口时，您的程序不会进行显示输出。但用户可以切换到其他程序，并在那里进行其他工作，而后台打印程序则能继续将输出文件发送到打印机。

程序 13-6 所示的 PRINT2 程序在 PRINT1 中添加了一个中止过程和必要的支持——调用 AbortProc 函数并调用 EnableWindow 两次（第一次禁用窗口，第二次启用窗口）。

程序 13-6 PRINT2 程序

PRINT2.C

```
/*-------------------------
    PRINT2.C——Printing with Abort Procedure
              (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

HDC GetPrinterDC (void) ;              // in GETPRNDC.C
void PageGDICalls (HDC, int, int) ;    // in PRINT.C

HINSTANCE  hInst ;
TCHAR      szAppName[] = TEXT ("Print2") ;
TCHAR      szCaption[] = TEXT ("Print Program 2 (Abort Procedure)") ;

BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
     MSG msg ;

     while (PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return TRUE ;
}

BOOL PrintMyPage (HWND hwnd)
{
     static DOCINFO  di = { sizeof (DOCINFO), TEXT ("Print2: Printing") } ;
     BOOL            bSuccess = TRUE ;
     HDC             hdcPrn ;
     short           xPage, yPage ;

     if (NULL == (hdcPrn = GetPrinterDC ()))
         return FALSE ;

     xPage = GetDeviceCaps (hdcPrn, HORZRES) ;
     yPage = GetDeviceCaps (hdcPrn, VERTRES) ;

     EnableWindow (hwnd, FALSE) ;

     SetAbortProc (hdcPrn, AbortProc) ;
```

```
    if (StartDoc (hdcPrn, &di) > 0)
    {
        if (StartPage (hdcPrn) > 0)
        {
            PageGDICalls (hdcPrn, xPage, yPage) ;

            if (EndPage (hdcPrn) > 0)
                EndDoc (hdcPrn) ;
            else
                bSuccess = FALSE ;
        }
    }
    else
        bSuccess = FALSE ;

    EnableWindow (hwnd, TRUE) ;
    DeleteDC (hdcPrn) ;
    return bSuccess ;
}
```

13.2.5　添加打印对话框

PRINT2 还不能令人十分满意。首先，这个程序没有直接指示出何时开始打印和何时结束打印。只有将鼠标指向程序，并且发现它没有反应时，才能断定它仍然在处理 PrintMyPage 例程。PRINT2 在进行后台处理时也没有给用户提供取消打印作业的机会。

您可能注意到，大多数 Windows 程序都为用户提供了一个取消当前正在进行的打印操作的机会。一个小的对话框出现在屏幕上，它显示一些说明文本和 Cancel 按钮。在 GDI 将打印输出保存到磁盘文件或（如果禁用后台程序）打印机正在打印的整个期间，程序都显示这个对话框。它是一个非模态对话框，必须提供对话过程。

通常称这个对话框为“放弃对话框”，称这种对话过程为“放弃对话过程”。为了更清楚地把它和“放弃过程”区别开来，我们称这种对话过程为“打印对话过程”。放弃过程（名为“AbortProc”）和打印对话过程（将命名为“PrintDlgProc”）是两个不同的输出函数。如果想以一种很专业的 Windows 方式进行打印，就必须拥有这两个函数。

这两个函数的交互作用方式如下：必须对 AbortProc 中的 PeekMessage 循环进行修改，以便将消息发送给非模态对话框的对话框窗口过程。PrintDlgProc 必须处理 WM_COMMAND 消息，以检查 Cancel 按钮的状态。如果 Cancel 钮被按下，就将一个 bUserAbort 全局变量置为 TRUE。AbortProc 返回的值正好和 bUserAbort 相反。您可能还记得，如果 AbortProc 返回 TRUE，那么继续打印；若返回 FALSE，则放弃打印。在 PRINT2 中，我们总是返回 TRUE。现在，用户在打印对话框中按下 Cancel 按钮时将返回 FALSE。程序 13-7 所示的 PRINT3 程序实现了这个逻辑。

程序 13-7　PRINT3 程序

PRINT3.C

```
/*--------------------
    PRINT3.C——Printing with Dialog Box
              (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>

HDC GetPrinterDC (void) ;                  // in GETPRNDC.C
void PageGDICalls (HDC, int, int) ;        // in PRINT.C

HINSTANCE  hInst ;
TCHAR      szAppName[] = TEXT ("Print3") ;
TCHAR      szCaption[] = TEXT ("Print Program 3 (Dialog Box)") ;

BOOL bUserAbort ;
HWND hDlgPrint ;

BOOL CALLBACK PrintDlgProc (HWND hDlg, UINT message,
                            WPARAM wParam, LPARAM lParam)
{
     switch (message)
     {
     case WM_INITDIALOG:
          SetWindowText (hDlg, szAppName) ;
          EnableMenuItem (GetSystemMenu (hDlg, FALSE), SC_CLOSE, MF_GRAYED) ;
          return TRUE ;

     case WM_COMMAND:
          bUserAbort = TRUE ;
          EnableWindow (GetParent (hDlg), TRUE) ;
          DestroyWindow (hDlg) ;
          hDlgPrint = NULL ;
          return TRUE ;
     }
     return FALSE ;
}

BOOL CALLBACK AbortProc (HDC hdcPrn, int iCode)
{
     MSG msg ;

     while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
     {
```

```
            if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))
            {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
            }
        }
        return !bUserAbort ;
}

BOOL PrintMyPage (HWND hwnd)
{
        static DOCINFO  di = { sizeof (DOCINFO), TEXT ("Print3: Printing") } ;
        BOOL            bSuccess = TRUE ;
        HDC             hdcPrn ;
        int             xPage, yPage ;

        if (NULL == (hdcPrn = GetPrinterDC ()))
            return FALSE ;

        xPage = GetDeviceCaps (hdcPrn, HORZRES) ;
        yPage = GetDeviceCaps (hdcPrn, VERTRES) ;

        EnableWindow (hwnd, FALSE) ;

        bUserAbort = FALSE ;
        hDlgPrint = CreateDialog (hInst, TEXT ("PrintDlgBox"),
                                  hwnd, PrintDlgProc) ;

        SetAbortProc (hdcPrn, AbortProc) ;

        if (StartDoc (hdcPrn, &di) > 0)
        {
            if (StartPage (hdcPrn) > 0)
            {
                PageGDICalls (hdcPrn, xPage, yPage) ;

                if (EndPage (hdcPrn) > 0)
                    EndDoc (hdcPrn) ;
                else
                    bSuccess = FALSE ;
            }
        }
        else
            bSuccess = FALSE ;

        if (!bUserAbort)
        {
            EnableWindow (hwnd, TRUE) ;
            DestroyWindow (hDlgPrint) ;
```

```
    }

    DeleteDC (hdcPrn) ;

    return bSuccess && !bUserAbort ;
}
```

PRINT.RC (摘录)

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog

PRINTDLGBOX DIALOG DISCARDABLE 20, 20, 186, 63
STYLE DS_MODALFRAME | WS_POPUP | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Cancel",IDCANCEL,67,42,50,14
    CTEXT           "Cancel Printing",IDC_STATIC,7,21,172,8
END
```

如果使用 PRINT3,那么最好临时禁用后台打印;否则,只有在后台程序从 PRINT3 中接收数据时才可见的 Cancel 按钮可能会很快消失,让您根本没有机会去按它。如果当您按 Cancel 按钮时打印并不立即终止(特别是在一个慢速打印机上),不要惊奇。打印机有一个内部缓冲区,在打印机终止之前其中的数据必须全部送出,按 Cancel 只是告诉 GDI 不要向打印机的缓冲区发送更多的数据而已。

PRINT3 增加了两个全局变量:一个是叫做"bUserAbort"的布尔变量,另一个是叫做"hDlgPrint"的对话框窗口句柄。PrintMyPage 函数将 bUserAbort 初始化为 FALSE。与 PRINT2 一样,程序的主窗口是禁用的。指向 AbortProc 的指针用于 SetAbortProc 调用中,而指向 PrintDlgProc 的指针用于 CreateDialog 调用中。CreateDialog 返回的窗口句柄保存在 hDlgPrint 中。

现在,AbortProc 中的消息循环如下:

```
while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
{
    if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
}
return !bUserAbort ;
```

仅当 bUserAbort 为 FALSE，也就是用户还没有终止打印工作时，才调用 PeekMessage。IsDialogMessage 函数用来将消息发送给非模态对话框。和普通的非模态对话框一样，对话框窗口的句柄在这个调用之前受到检查。AbortProc 的返回值正好与 bUserAbort 相反。开始时，bUserAbort 为 FALSE，因此 AbortProc 返回 TRUE，表示继续进行打印，但是 bUserAbort 可能在打印对话过程中被设置为 TRUE。

PrintDlgProc 函数是相当简单的。当处理 WM_INITDIALOG 时，该函数将窗口标题设置为程序名，并且禁用系统菜单上的 Close 选项。如果用户按下 Cancel 按钮，PrintDlgProc 将接收到 WM_COMMAND 消息：

```
case WM_COMMAND :
     bUserAbort = TRUE ;
     EnableWindow (GetParent (hDlg), TRUE) ;
     DestroyWindow (hDlg) ;
     hDlgPrint = NULL ;
     return TRUE ;
```

将 bUserAbort 设置为 TRUE，则说明用户已经决定取消打印操作，主窗口被启动，而对话框被清除（按顺序完成这两项活动是很重要的，否则，运行在 Windows 中的其他一些程序将变成活动程序，而您的程序将消失到背景中）。与通常情况一样，将 hDlgPrint 设置为 NULL，防止在消息循环中调用 IsDialogMessage。

仅当 AbortProc 用 PeekMessage 检索到消息，并用 IsDialogMessage 将它们传送给对话框窗口过程时，这个对话框才接收消息。仅当 GDI 模块处理 EndPage 函数时，才调用 AbortProc。如果 GDI 发现 AbortProc 的返回值是 FALSE，它将控制从 EndPage 返回到 PrintMyPage。它不返回错误码。至此，PrintMyPage 认为页已经发完了，并调用 EndDoc 函数。但是，由于 GDI 模块还没有完成对 EndPage 调用的处理，所以不会打印出什么东西来。

某些清除工作尚待完成。如果用户没在对话框中取消打印作业，那么仍然显示对话框。PrintMyPage 重新启用它的主窗口并清除对话框：

```
if (!bUserAbort)
{
     EnableWindow (hwnd, TRUE) ;
     DestroyWindow (hDlgPrint) ;
}
```

两个变量会通知您发生了什么事：bUserAbort 可以告诉您用户是否终止了打印作业，bSuccess 会告诉您是否出了故障，您可以用这些变量来完成想做的工作。PrintMyPage 只简单地对它们进行逻辑与运算，把值返回给 WndProc：

```
return bSuccess && !bUserAbort ;
```

13.2.6 为 POPPAD 添加打印功能

现在准备在 POPPAD 程序中添加打印功能，并且宣布 POPPAD 已告完毕。这需要第十一章中的各种 POPPAD 文件，此外，还需要程序 13-8 中的 POPPRNT.C 文件。

程序 13-8 为 POPPAD 添加打印功能的 POPPRNT.C 文件

POPPRNT.C

```
/*-----------------------
    POPPRNT.C——Popup Editor Printing Functions
   -----------------------*/

#include <windows.h>
#include <commdlg.h>
#include "resource.h"

BOOL bUserAbort ;
HWND hDlgPrint ;

BOOL CALLBACK PrintDlgProc (HWND hDlg, UINT msg, WPARAM wParam, LPARAM lParam)
{
     switch (msg)
     {
     case WM_INITDIALOG :
          EnableMenuItem (GetSystemMenu (hDlg, FALSE), SC_CLOSE, MF_GRAYED) ;
          return TRUE ;

     case WM_COMMAND :
          bUserAbort = TRUE ;
          EnableWindow (GetParent (hDlg), TRUE) ;
          DestroyWindow (hDlg) ;
          hDlgPrint = NULL ;
          return TRUE ;
     }
     return FALSE ;
}

BOOL CALLBACK AbortProc (HDC hPrinterDC, int iCode)
{
     MSG msg ;

     while (!bUserAbort && PeekMessage (&msg, NULL, 0, 0, PM_REMOVE))
     {
          if (!hDlgPrint || !IsDialogMessage (hDlgPrint, &msg))
          {
```

```
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
            }
        }
        return !bUserAbort ;
}

BOOL PopPrntPrintFile (HINSTANCE hInst, HWND hwnd, HWND hwndEdit,
                       PTSTR szTitleName)
{
        static DOCINFO  di = { sizeof (DOCINFO) } ;
        static PRINTDLG pd ;
        BOOL            bSuccess ;
        int             yChar, iCharsPerLine, iLinesPerPage, iTotalLines,
                        iTotalPages, iPage, iLine, iLineNum ;
        PTSTR           pstrBuffer ;
        TCHAR           szJobName [64 + MAX_PATH] ;
        TEXTMETRIC      tm ;
        WORD            iColCopy, iNoiColCopy ;

            // Invoke Print common dialog box

        pd.lStructSize          = sizeof (PRINTDLG) ;
        pd.hwndOwner            = hwnd ;
        pd.hDevMode             = NULL ;
        pd.hDevNames            = NULL ;
        pd.hDC                  = NULL ;
        pd.Flags                = PD_ALLPAGES | PD_COLLATE |
                                  PD_RETURNDC | PD_NOSELECTION ;
        pd.nFromPage            = 0 ;
        pd.nToPage              = 0 ;
        pd.nMinPage             = 0 ;
        pd.nMaxPage             = 0 ;
        pd.nCopies              = 1 ;
        pd.hInstance            = NULL ;
        pd.lCustData            = 0L ;
        pd.lpfnPrintHook        = NULL ;
        pd.lpfnSetupHook        = NULL ;
        pd.lpPrintTemplateName  = NULL ;
        pd.lpSetupTemplateName  = NULL ;
        pd.hPrintTemplate       = NULL ;
        pd.hSetupTemplate       = NULL ;

        if (!PrintDlg (&pd))
            return TRUE ;

        if (0 == (iTotalLines = SendMessage (hwndEdit, EM_GETLINECOUNT, 0, 0)))
            return TRUE ;
```

```
    // Calculate necessary metrics for file

GetTextMetrics (pd.hDC, &tm) ;
yChar = tm.tmHeight + tm.tmExternalLeading ;

iCharsPerLine = GetDeviceCaps (pd.hDC, HORZRES) / tm.tmAveCharWidth ;
iLinesPerPage = GetDeviceCaps (pd.hDC, VERTRES) / yChar ;
iTotalPages   = (iTotalLines + iLinesPerPage - 1) / iLinesPerPage ;

    // Allocate a buffer for each line of text

pstrBuffer = malloc (sizeof (TCHAR) * (iCharsPerLine + 1)) ;

    // Display the printing dialog box

EnableWindow (hwnd, FALSE) ;

bSuccess  = TRUE ;
bUserAbort = FALSE ;

hDlgPrint = CreateDialog (hInst, TEXT ("PrintDlgBox"),
                          hwnd, PrintDlgProc) ;

SetDlgItemText (hDlgPrint, IDC_FILENAME, szTitleName) ;
SetAbortProc (pd.hDC, AbortProc) ;

    // Start the document

GetWindowText (hwnd, szJobName, sizeof (szJobName)) ;
di.lpszDocName = szJobName ;
if (StartDoc (pd.hDC, &di) > 0)
{
        // Collation requires this loop and iNoiColCopy

    for (iColCopy = 0 ;
         iColCopy < ((WORD) pd.Flags & PD_COLLATE ? pd.nCopies : 1) ;
         iColCopy++ )
    {
        for (iPage = 0 ; iPage < iTotalPages ; iPage++ )
        {
            for (iNoiColCopy = 0 ;
                 iNoiColCopy < (pd.Flags & PD_COLLATE ? 1 : pd.nCopies);
                 iNoiColCopy++ )
            {
                    // Start the page

                if (StartPage (pd.hDC) < 0)
```

```
                    {
                        bSuccess = FALSE ;
                        break ;
                    }

                    // For each page, print the lines

                    for (iLine = 0 ; iLine < iLinesPerPage ; iLine++)
                    {
                        iLineNum = iLinesPerPage * iPage + iLine ;

                        if (iLineNum > iTotalLines)
                            break ;

                        *(int *) pstrBuffer = iCharsPerLine ;

                        TextOut (pd.hDC, 0, yChar * iLine, pstrBuffer,
                                 (int) SendMessage (hwndEdit, EM_GETLINE,
                                 (WPARAM) iLineNum, (LPARAM) pstrBuffer));
                    }

                    if (EndPage (pd.hDC) < 0)
                    {
                        bSuccess = FALSE ;
                        break ;
                    }

                    if (bUserAbort)
                        break ;
                }

                if (!bSuccess || bUserAbort)
                    break ;
            }

            if (!bSuccess || bUserAbort)
                break ;
        }
    }
    else
        bSuccess = FALSE ;

    if (bSuccess)
        EndDoc (pd.hDC) ;

    if (!bUserAbort)
    {
        EnableWindow (hwnd, TRUE) ;
```

```
        DestroyWindow (hDlgPrint) ;
    }

    free (pstrBuffer) ;
    DeleteDC (pd.hDC) ;

    return bSuccess && !bUserAbort ;
}
```

与 POPPAD 尽量利用 Windows 高级特性来简化程序的方针一致，POPPRNT.C 文件演示了使用 PrintDlg 函数的方法。这个函数包括在通用对话框库中，并使用一个 PRINTDLG 类型的结构。

通常，一个 Print 选项被包含在程序的 File 菜单中。当用户选中 Print 选项时，程序可以初始化 PRINTDLG 结构的域，并调用 PrintDlg。

PrintDlg 显示一个对话框，它允许用户选择打印页的范围。因此，这个对话框特别适用于像 POPPAD 这样能打印多页文档的程序。这种对话框同时也给出了一个确定副本份数的编辑区和名为“Collate(归类)”的复选框。“归类”影响着多个副本页的顺序。例如，如果文档是 3 页，用户要求打印三份副本，则这个程序能以两种顺序之一打印它们。归类后的副本的页码顺序为 1、2、3、1、2、3、1、2、3，未经归类的副本的页码顺序是 1、1、1、2、2、2、3、3、3。程序的责任是以正确的顺序打印副本。

这个对话框也允许用户选择非默认打印机，它包括一个标记为 Properties 的按钮，可以激活设备方式对话框。这样，至少允许用户选择纵向或横向方式。

从 PrintDlg 函数返回后，PRINTDLG 结构的字段指明打印页的范围和是否对多个副本进行归类。这个结构同时也给出了准备使用的打印机设备描述表句柄。

在 POPPRNT.C 中，PopPrntPrintFile 函数(当用户在 File 菜单里选中 Print 选项时，它由 POPPAD 调用)调用 PrintDlg，然后开始打印文件。PopPrntPrintFile 完成某些计算，以确定一行能容纳多少字符和一页能容纳多少行。这个过程涉及到调用 GetDeviceCaps 来确定页的分辨率，调用 GetTextMetrics 来确定字符的大小。

这个程序通过将一个 EM_GETLINECOUNT 消息发送给编辑控制来获取文档中的总行数(在变量 iTotalLines 中)。保存各行内容的缓冲区分配在局部内存中。对每一行，缓冲区的第一个字被设置为该行中字符数。把 EM_GETLINE 消息发送给编辑控制可以把一行复制到缓冲区中，然后用 TextOut 把这一行发送到打印机设备描述表中。(POPPRNT.C 还没有聪明到对超出打印宽度的行进行换行。在第十七章我们将讨论这种换行技术。)

为了确定副本份数，应注意打印文本的逻辑包括两个 for 循环。第一个 for 循环使用了一个叫做“iColCopy”的变量，当用户指定将副本归类时，它将起作用。第二个 for 循环使用了一个叫做“iNonColCopy”的变量，当不对副本进行归类时，它将起作用。

如果 StartPage 或 EndPage 返回一个错误或者 bUserAbort 为 TRUE，那么这个程序退出增加页号的那个 for 循环。如果中止过程的返回值是 FALSE，则 EndPage 不返回错误。正是由于这个原因，在下一页开始之前，要直接测试 bUserAbort。如果没有报告错误，则进行 EndDoc 调用：

```
if (!bError)
    EndDoc (hdcPrn) ;
```

您可能想通过打印多页文件来测试 POPPAD。您可以从打印任务窗口中监视打印进展情况。在 GDI 处理完第一个 EndPage 调用之后，首先打印的文件将显示在打印任务窗口中。此时，后台打印程序开始把文件发送到打印机。然后，如果在 POPPAD 中取消打印作业，那么后台打印程序将终止打印，这也就是中止过程返回 FALSE 的结果。当文件出现在打印任务窗口中，您也可以通过从 Document 菜单中选择 Cancel Printing 来取消打印作业，在这种情况下，POPPAD 中的 EndPage 调用返回一个错误。

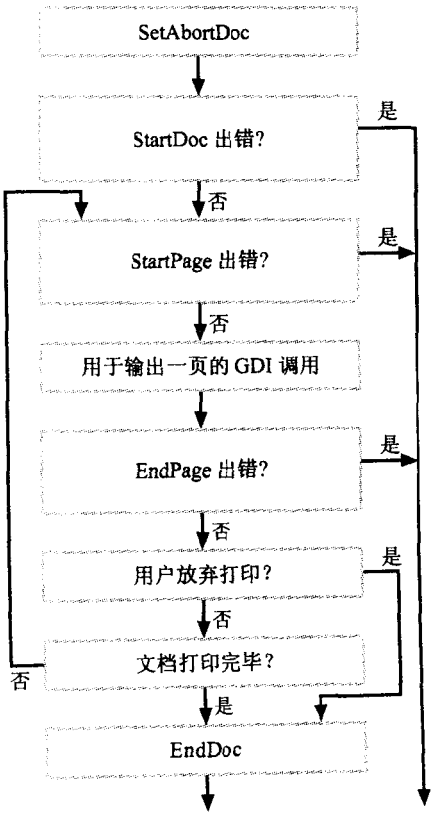


图 13-3　用于多页打印的函数调用顺序

Windows 程序设计的新手经常会抱住 AbortDoc 函数不放，但实际上这个函数几乎不在打印中使用。像在 POPPAD 中看到的那样，用户几乎随时都可以取消打印作业，或者通过 POPPAD 的打印对话框或者通过打印任务窗口。这两种方法都不需要程序使用 AbortDoc 函数。POPPAD 中允许 AbortDoc 的唯一时刻是在对 StartDoc 的调用和对 EndPage 的第一个调用之间，但是程序很快就会执行过去，以至不再需要 AbortDoc。

图 13-3 显示出正确打印多页文档的打印函数的调用顺序。检查 bUserAbort 的值是否为 TRUE 的最佳位置是在每个 EndPage 函数之后。仅当对先前的打印函数的调用没有产生错误时，才使用 EndDoc 函数。实际上，如果任何一个打印函数的调用出现错误，那么表演就结束了，您也可以回家了。

第十四章

位图和 Bitblt

位图是一个二维的位数组，它与图像的像素一一对应。当现实世界的图像捕获进位图以后，图像被分割成网格，并以像素作为取样单位。在位图中的每个像素值指明了一个单位网格内图像的平均颜色。单色位图每个像素只需要一位，灰色或彩色位图中每个像素需要多个位。

位图代表了 Windows 程序中存储图像信息的两种方法之一。存储图像信息的另一种形式是元文件，我将在第十八章讨论。元文件是对图像的描述而不是数字化表示。

以后我将更详细地讨论，Microsoft Windows 3.0 定义了一种被称为“设备无关的位图(DIB:device-independent bitmap)”。我将在下一章讨论 DIB。本章主要讨论 GDI 位图对象，它是一种在 Windows 内先于 DIB 实现的位图。在本章大量的示例程序中，Windows 先于 DIB 实现的位图支持仍然是相当有用的。

14.1 位图基础

位图和元文件在计算机图形里都有它们自己的位置。位图经常用来表示来自真实世界的复杂图像，例如数字化的照片或者视频图像。元文件更适合于描述由人或者机器生成的图像，比如建筑图纸。位图和元文件都能存于内存或作为文件存于磁盘上，并且都能通过剪贴板在 Windows 应用程序之间传输。

位图和元文件的区别在于光栅图像和矢量图像之间的差别。光栅图像用离散的像素来处理输出设备；而矢量图像用笛卡尔坐标系统来处理输出设备，可在其中绘制线和填充对象。现在大多数的图像输出设备是光栅设备，包括视频显示器、点阵打印机、激光打印机，以及喷墨打印机。而笔式绘图仪则是矢量输出设备。

位图有两个主要的缺点。第一个问题是容易受设备依赖性的影响。最明显的设备依赖是颜色。在单色设备上显示彩色位图总是不能令人满意的。另一个问题是位图经

常暗示了特定的显示分辨率和图像纵横比。尽管位图能被拉伸和压缩，但是此过程通常包括复制或删除像素的某些行和列，这样会导致缩放后的图像出现失真。元文件几乎能缩放到任意尺寸而不失真。

位图的第二个主要缺点是需要的存储空间很大。例如，描述完整的 640 像素 × 480 像素，16 色的视频图形阵列（VGA：Video Graphics Array）屏幕的一幅位图需要大于 150 KB 的空间；一幅 1024 × 768，且每个像素为 24 位颜色的图像则需要大于 2 MB 的空间。元文件通常需要比位图更少的存储空间。位图的存储空间由图像的大小及其包含的颜色数决定，而元文件的存储空间则由图像的复杂程度及其包含的单独的 GDI 指令数决定。

然而，位图优于元文件之处在于速度。将位图复制给视频显示器通常比复制图元文件的速度要快得多。最近几年，压缩技术允许压缩位图尺寸，以使它能有效地通过电话线传输，并广泛用于 Internet 上的 World Wide Web 页。

14.1.1　位图的来源

位图可以手工创建，例如使用 Windows 98 附带的“画图”程序。一些人使用光栅绘图软件而不是矢量绘图软件，他们工作在假设的环境中：最终的图像过于复杂，以至于不能用线和填充区域来表达。

位图图像也能由计算机代码计算生成。尽管大多数计算生成的图像能按矢量图形元文件保存，但是高清晰度的表面通常还需要位图。

现在，位图通常用于描述真实世界的图像，并且有许多硬件设备能让您把现实世界的图像输入到计算机。这类硬件通常使用电荷耦合装置（CCD：charge-coupled device），这种装置接触到光就释放电荷。某些情况下 CCD 单元排列成一组，一个像素对应一个 CCD；为节约开支，只用一行 CCD 扫描图像。

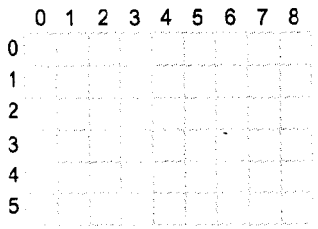
在这些计算机 CCD 设备中，扫描仪是最古老的。它用一行 CCD 沿着打印图像（如照片）的表面扫描。CCD 根据光的强度产生电荷。模数转换器（ADC：Analog-to-digital converters）把电荷转换为数字，然后排列成位图。

便携式视频摄像机也利用 CCD 单元组来捕获图像。通常，这些图像记录在录像带上。然而，视频输出能直接进入视频帧捕获器（frame grabber），该装置能把模拟视频信号转换为一组像素值。这些帧捕获器与任何兼容的视频源都能同时使用，例如 VCR、光盘、DVD 播放机，或电缆解码箱。

最近，数字照相机对于家庭用户来说是可以接受的了。它看起来与普通的照相机非常相像，但它不用胶片，而是用一组 CCD 来捕获图像，并且通过内置 ADC 把数字图像直接存储到照相机的内存中。通常，数字照相机要通过串行端口与计算机交换数据。

14.2 位图尺寸

位图呈矩形，并有空间尺寸，图像的高度和宽度都以像素为单位。例如，下面的网格可代表一个很小的位图：宽度为 9 像素，高度为 6 像素，或者简单地表示为 9×6：



习惯上，位图的速记尺寸是先给出宽度。这个位图总共有 9×6 即 54 个像素。我将经常使用符号 cx 和 cy 来表示位图的宽度和高度。c 表示记数，因此 cx 和 cy 是沿着 x 轴（水平）和 y 轴（垂直）的像素数。

我们能根据 x 和 y 坐标来描述位图上具体的像素。一般地（并不都是这样），在网格内计算像素时，位图原点位于图像的左上角。这样，在此位图右下角的像素坐标就是 (8, 5)。因为从 0 开始计数，所以此值比图像的宽度和高度小 1。

位图的空间尺寸通常也指定了分辨率，但这是一个有争议的词。我们说我们的视频显示有 640×480 的分辨率，而激光打印机的分辨率是每英寸 300 点。我认为这个术语的后一种用法是正确的，分辨率指的是单位长度内像素的数量。在这种意义上位图的分辨率指的是位图在单位长度内的像素数。不管怎样，当我使用分辨率这个词时，其含义可根据上下文确定。

位图是矩形的，但是计算机内存是线性的。通常（但不都是这样）位图按行存储在内存中，且从顶行像素开始到底行结束。（DIB 是此规则的一个主要的例外。）每一行，像素都从最左边的像素开始，依次向右存储。这就像存储多行文本中的字符一样。

14.2.1 颜色和位图

除空间度量单位以外，位图还有颜色度量单位。这指的是每个像素所需要的位数，有时也称为位图的颜色深度（color depth）、位数（bit-count），或位/每像素（bpp：bits per pixel）。位图中的每个像素都有相同数量的颜色位。

每像素 1 位的位图称为二级（bilevel）、二色（bicolor）或单色（monochrome）位图。每像素可以是 0 或 1，0 可以表示黑色，1 可以表示白色，但并不总是这样。对于其他颜色，一个像素就需要有多个位。可能的颜色数等于 2^i（i 为位数）。用 2 位可以得到 4 种颜色，用 4 位可以得 16 种颜色，8 位可得到 256 种颜色，16 位可得到 65 536 种颜色，而 24

位可得到16 777 216种颜色。

如何将颜色位的组合与人们所熟识的颜色相对应是处理位图时经常出现(通常都是麻烦的)的问题。

14.2.2 现实世界的设备

位图可按其颜色位数来分类。在 Windows 的发展过程中,不同的位图颜色格式取决于常用视频显示适配器的功能。实际上,我们可把视频显示内存看作是一幅巨大的位图——从显示器上就可以看见。

Windows 1.0 采用的多数视频适配器是 IBM 的彩色图像适配器(CGA:Color Graphics Adapter)和大力神图形卡(HGC:Hercules Graphics Card)。HGC 是单色设备,因而 CGA 不得不用于 Windows 的单色图形模式下。单色位图现在还很常用(例如,鼠标的光标一般为单色),而且单色位图除显示图像以外还有其他用途。

随着增强图形适配器(EGA:Enhanced Graphics Adapter)的介入,Windows 用户开始接触 16 色。每个像素需要 4 个颜色位。(实际上,EGA 比这里所讲的更复杂,它还包括一个 64 种颜色的调色板,应用程序可以从中选择任意的 16 种颜色,但 Windows 只按较简单的方法使用 EGA。)在 EGA 中使用的 16 种颜色是黑、白、两种灰色、暗和亮版本的红、绿、蓝(三种最初添加的颜色)、青(蓝和绿组合的颜色)、品红(蓝和红组合的颜色),以及黄色(红和绿组合的颜色)。现在认为这 16 种颜色是 Windows 的最低标准。同样,其他 16 色位图也可以在 Windows 中显示。大多数的图标都是 16 色的位图。通常,简单的卡通图像也可以用这 16 种颜色制作。

在 16 色位图中的颜色编码有时称为 IRGB(高亮红绿蓝:Intensity-Red-Green-Blue),并且实际上是来源于 IBM CGA 字符模式下最初使用的颜色。每个像素所用的 4 个 IRGB 颜色位都映射为下页表 14-1 所示的 Windows 十六进制 RGB 颜色。

EGA 的内存组成了 4 个"颜色板",也就是说,定义每个像素颜色的 4 位在内存中是不连续的。然而,这样组织显示内存便于使所有的亮度位在一起、所有的红色位在一起,等等。这样听起来类似于设备依赖特性,即 Windows 程序员不需要了解所有细节,但这时应或多或少地知道一些。不过,这些颜色板会出现在一些 API 调用中,例如 GetDeviceCaps 和 CreateBitmap。

Windows 98 和 Microsoft Windows NT 需要 VGA 或分辨率更高的图形卡。这是 Windows 目前所能接受的最低标准视频图形卡。

1987 年,IBM 引入了最初的视频图像阵列(VGA),以及 PS/2 的个人计算机。它提供了许多不同的视频模式,但最好的图像模式(Windows 也使用其中之一)是水平显示 640 个像素,垂直显示 480 个像素,带有 16 种颜色。要显示 256 种颜色,最初的 VGA 必须切换到 320×240 的图形模式,这个像素数不适合 Windows 的正常工作。

人们已经忘记了最初 VGA 板的颜色限制，因为其他硬件制造商很快就开发了“Super-VGA(SVGA)”适配器，它包括更多的视频内存，可显示 256 种颜色，并有高于 640×480 的模式。这是现在的标准，而且也是一件好事，因为 16 种颜色根本不足以显示现实世界中的图像。

表 14-1

IRGB	RGB 颜色	颜色名
0000	00-00-00	黑
0001	00-00-80	深蓝
0010	00-80-00	深绿
0011	00-80-80	深青
0100	80-00-00	深红
0101	80-00-80	深品红
0110	80-80-00	深黄
0111	C0-C0-C0	浅灰
1000	80-80-80	深灰
1001	00-00-FF	蓝
1010	00-FF-00	绿
1011	00-FF-FF	青
1100	FF-00-00	红
1101	FF-00-FF	品红
1110	FF-FF-00	黄
1111	FF-FF-FF	白

显示 256 种颜色的视频适配器模式每像素用 8 位表示。不过，这些 8 位值都不必与实际的颜色相符。事实上，视频板合并了“调色板查找表(palette lookup table)”，该表允许软件指定这 8 位值，以便与实际颜色相匹配。在 Windows 中，应用程序不能直接访问调色板查找表；实际上，Windows 保留了 256 种颜色中的 20 种，而应用程序可以通过“Windows 调色板管理程序”来自定义其余的 236 种颜色。关于这些内容，我将在第十六章详细介绍。调色板管理器允许应用程序在 256 色显示器上显示现实世界的位图。Windows 所保留的 20 种颜色如下页表 14-2 所示。

最近几年，真彩色视频适配器卡很普遍，它们在每像素使用 16 位或 24 位。有时每像素用 16 位，其中有 1 位不用，而其他 15 位分配给红、绿和蓝三原色。这样红、绿和蓝每种都有 32 级灰度，组合起来就可以达到 32 768 种颜色。更普遍的是，6 位用于绿色(人类对此颜色最敏感)，这样就可得到 65 536 种颜色。对于非技术性的 PC 用户来说，他们并不喜欢看到诸如 32 768 或 65 536 之类的数字，因此通常将这种视频显示卡称为“伪彩色”适配器，它能提供数以千计的颜色。

到了每个像素 24 位时，我们总共有了 16 777 216 种颜色(或者说真彩色、数以百万

计的颜色)。每个像素使用3个字节。这种标准可能会保持许多年,因为它几乎达到了人类感官的极限,而且也很方便。

表 14-2

颜色值	RGB 颜色	颜色名
00000000	00-00-00	黑
00000001	80-00-00	深红
00000010	00-80-00	深绿
00000011	80-80-00	深黄
00000100	00-00-80	深蓝
00000101	80-00-80	深品红
00000110	00-80-80	深青
00000111	C0-C0-C0	浅灰
00001000	C0-DC-C0	美元绿
00001001	A6-CA-F0	天蓝
11110110	FF-FB-F0	乳白
11110111	A0-A0-A4	中性灰
11111000	80-80-80	深灰
11111001	FF-00-00	红
11111010	00-FF-00	绿
11111011	FF-FF-00	黄
11111100	00-00-FF	蓝
11111101	FF-00-FF	品红
11111110	00-FF-FF	青
11111111	FF-FF-FF	白

在调用 GetDeviceCaps 时(参见第五章的 DEVCAPS 程序),您能利用 BITSPIXEL 和 PLANES 常数来获得视频卡的颜色结构,这些值如表 14-3 所示。

表 14-3

BITSPIXEL	PLANES	颜色数
1	1	2
1	4	16
8	1	256
15 或 16	1	32 768 或 65 536
24 或 32	1	16 777 216

而今,您不应再遇到单色视频显示器了,但即便遇见了,您的应用程序不应该产生

冲突。

14.2.3 GDI 支持的位图

Windows 图形设备接口(GDI:Graphics Device Interface)从 1.0 版开始支持位图。不过,到了 Windows 3.0 以前,Windows 下唯一支持 GDI 对象的只有位图,它用位图句柄来引用。这些 GDI 位图对象是单色的,或者与实际的图像输出设备(例如视频显示器)有相同的颜色单位。例如,与 16 色 VGA 兼容的位图有 4 个颜色板。问题是这些颜色位图不能保存,也不能用于颜色单位不同的图像输出设备(如每像素占 8 位就可以产生 256 种颜色的设备)。

从 Windows 3.0 开始,定义了一种新的位图格式,我们称之为"设备无关位图(device-independent bitmap)",或者"DIB"。DIB 包括自己的颜色表,其中显示了 RGB 颜色对应的像素位。DIB 能显示在任何光栅输出设备上。这里唯一的问题是 DIB 的颜色通常一定会转换成设备实际表现出来的颜色。

与 DIB 一块,Windows 3.0 还引入了"Windows 调色板管理程序",它允许程序从显示的 256 种颜色中自定义颜色。就像我们在第十六章所看到的那样,应用程序通常结合"调色板管理程序"来显示 DIB。

Microsoft 在 Windows 95(和 Windows NT 4.0)中扩展了 DIB 的定义,并且在 Windows 98(和 Windows NT 5.0)中再次扩展。这些扩展增加了"图像颜色管理器(ICM:Image Color Management)",并允许 DIB 更精确地指定图像所需要的颜色。我将在第十五章简要讨论 ICM。

不论 DIB 多么重要,在处理位图时,早期的 GDI 位图对象依然扮演了重要的角色。掌握位图的最好方法是按时间顺序接触资料,从 GDI 位图对象和位块传送的概念开始。

14.3 位块传送

前面提到过,您可以把整个视频显示器看作是一幅大位图。您在屏幕上见到的像素由存储在视频显示适配卡上内存中的位来描述。任何视频显示的矩形区域也都是一个位图,其大小是它所包含的行列数。

让我们通过将图像从视频显示的一个区域复制到另一个区域,开始我们在位图世界的旅行吧! 这就是强大的 BitBlt 函数的工作。

Bitblt(读作"bit blit")代表"位块传送(bit-block transfer)"。BLT 起源于一条汇编语言指令,该指令在 DEC PDP-10 上完成内存块传送。术语"bitblt"第一次用在图像上是在 Xerox 公司的 Palo Alto Research Center(PARC)设计的 SmallTalk 系统。在 SmallTalk 中,所有的图形输出操作都是基于 bitblt 的。程序员有时将 blt 用作动词,例如:"Then I wrote some code to blt the happy face to the screen and play a wave file."

BitBlt 函数是像素移动程序，或者(更生动地)是一个光栅狂飙。您将看到，术语“传送(transfer)”与 BitBlt 函数不尽相同。此函数实际上对像素执行了一次位运算操作，并能产生一些有趣的结果。

14.3.1 简单的 BitBlt

程序 14-1 所示的 BITBLT 程序用 BitBlt 函数将程序的系统菜单图标(位于程序窗口的左上角)复制到它的客户区。

程序 14-1　BITBLT 程序

BITBLT.C

```
/*------------------
   BITBLT.C——BitBlt Demonstration
             (c) Charles Petzold, 1998
   ------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName [] = TEXT ("BitBlt") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_INFORMATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
```

```
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("BitBlt Demo"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static int      cxClient, cyClient, cxSource, cySource ;
    HDC             hdcClient, hdcWindow ;
    int             x, y ;
    PAINTSTRUCT     ps ;

    switch (message)
    {
    case WM_CREATE:
        cxSource = GetSystemMetrics (SM_CXSIZEFRAME) +
                   GetSystemMetrics (SM_CXSIZE) ;

        cySource = GetSystemMetrics (SM_CYSIZEFRAME) +
                   GetSystemMetrics (SM_CYCAPTION) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_PAINT:
        hdcClient = BeginPaint (hwnd, &ps) ;
        hdcWindow = GetWindowDC (hwnd) ;

        for (y = 0 ; y < cyClient ; y += cySource)
        for (x = 0 ; x < cxClient ; x += cxSource)
```

```
            {
                BitBlt (hdcClient, x, y, cxSource, cySource,
                        hdcWindow, 0, 0, SRCCOPY) ;
            }

            ReleaseDC (hwnd, hdcWindow) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
    }
```

但为什么只需要一个 BitBlt 函数呢？实际上，BITBLT 用系统菜单图标的多个副本来填充客户区（在此情况下是信息框中通常使用的 IDI_INFORMATION 图标），如图 14-1 所示。

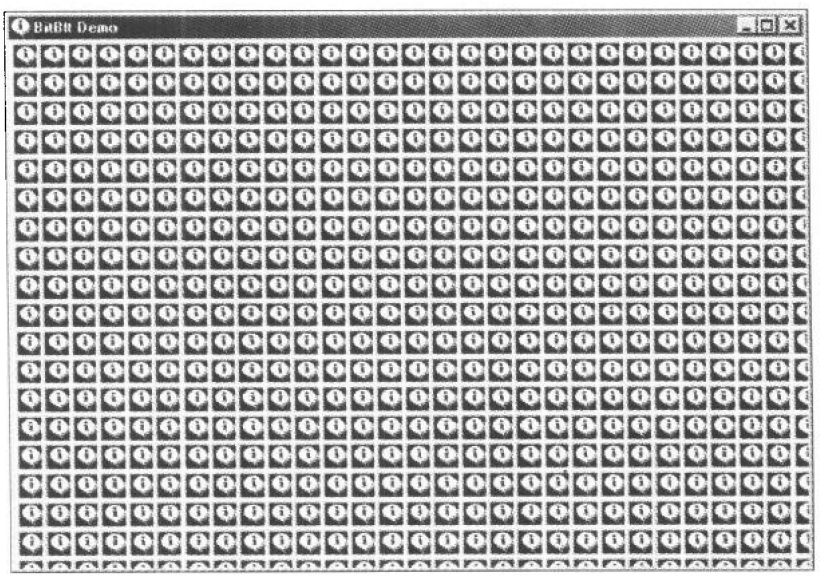


图 14-1　BITBLT 的屏幕显示

BitBlt 函数从称为“源”的设备描述表中将一个矩形区的像素传送到称为“目标”的另一个设备描述表中相同大小的矩形区。此函数的语法如下：

```
BitBlt (hdcDst, xDst, yDst, cx, cy, hdcSrc, xSrc, ySrc, dwROP) ;
```

源和目标设备描述表可以相同。

在 BITBLT 程序中,目标设备描述表是窗口的客户区,设备描述表句柄从 BeginPaint 函数获得。源设备描述表是应用程序的整个窗口,此设备描述表句柄是从 GetWindowDC 获得的。很明显,这两个设备描述表指的是同一个物理设备(视频显示器)。不过,这两个设备描述表的坐标原点不同。

xSrc 和 ySrc 参数指明了源图像左上角的坐标位置。在 BITBLT 中,这两个参数都设为 0,表明图像从源设备描述表(也就是整个窗口)的左上角开始,cx 和 cy 参数是图像的宽度和高度。BITBLT 根据从 GetSytemMetrics 函数获得的信息来计算这些值。

xDst 和 yDst 参数表明了复制图像位置左上角的坐标位置。在 BITBLT 中,这两个参数设置为不同的值,以便多次复制图像。对于第一次 BitBlt 调用,这两个参数设置为 0,将图像复制到客户区的左上角位置。

BitBlt 的最后一个参数是光栅操作符。我将在后面简要地介绍一下这个值。

请注意,BitBlt 是从实际视频显示内存传送像素,而不是从系统菜单图标的其他图像传送。如果移动 BITBLT 窗口,以使部分系统菜单图标移出屏幕,然后调整 BITBLT 窗口的尺寸使其重绘,这时您将发现 BITBLT 客户区中显示的是菜单图标的一部分。BitBlt 函数不再访问整个图像。

在 BitBlt 函数中,源和目标设备描述表可以相同。您可以重新编写 BITBLT,以使 WM_PAINT 处理执行以下内容:

```
BitBlt (hdcClient, 0, 0, cxSource, cySource,
     hdcWindow, 0, 0, SRCCOPY) ;

for (y = 0 ; y < cyClient ; y += cySource)
    for (x = 0 ; x < cxClient ; x += cxSource)
    {
        if (x > 0 || y > 0)
            BitBlt (hdcClient, x, y, cxSource, cySource,
                 hdcClient, 0, 0, SRCCOPY) ;
}
```

这将与前面显示的 BITBLT 一样产生相同的效果,只是客户区左上角比较暗。

在 BitBlt 内的最大限制是两个设备描述表必须兼容。这意味着或者其中之一必须是单色的,或者两者的每个像素都具有相同的位数。总而言之,您不能用此方法将屏幕上的某些图形复制到打印机。

14.3.2 拉伸位图

在 BitBlt 函数中,目标图像与源图像的尺寸是相同的,因为函数只有两个参数来说明宽度和高度。如果您想在复制时拉伸或者压缩图像尺寸,可以使用 StretchBlt 函数。

StretchBlt 函数的语法如下：

```
StretchBlt (hdcDst, xDst, yDst, cxDst, cyDst,
        hdcSrc, xSrc, ySrc, cxSrc, cySrc, dwROP) ;
```

此函数添加了两个参数。现在的函数就分别包含了目标和源各自的宽度和高度。STRETCH 程序演示了 StretchBlt 函数，如程序 14-2 所示。

程序 14-2　STRETCH 程序

STRETCH.C

```
/*-------------------
    STRETCH.C——StretchBlt Demonstration
                (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName [] = TEXT ("Stretch") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_INFORMATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }
```

```
     hwnd = CreateWindow (szAppName, TEXT ("StretchBlt Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static int        cxClient, cyClient, cxSource, cySource ;
     HDC               hdcClient, hdcWindow ;
     PAINTSTRUCT       ps ;

     switch (message)
     {
     case WM_CREATE:
          cxSource = GetSystemMetrics (SM_CXSIZEFRAME) +
                     GetSystemMetrics (SM_CXSIZE) ;

          cySource = GetSystemMetrics (SM_CYSIZEFRAME) +
                     GetSystemMetrics (SM_CYCAPTION) ;
          return 0 ;

     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_PAINT:
          hdcClient = BeginPaint (hwnd, &ps) ;
          hdcWindow = GetWindowDC (hwnd) ;

          StretchBlt (hdcClient, 0, 0, cxClient, cyClient,
                      hdcWindow, 0, 0, cxSource, cySource, MERGECOPY) ;

          ReleaseDC (hwnd, hdcWindow) ;
          EndPaint (hwnd, &ps) ;
          return 0 ;
```

```
    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

此程序只调用了 StretchBlt 函数一次，但通过调用此函数用系统菜单图标填充了整个客户区，如图 14-2 所示。

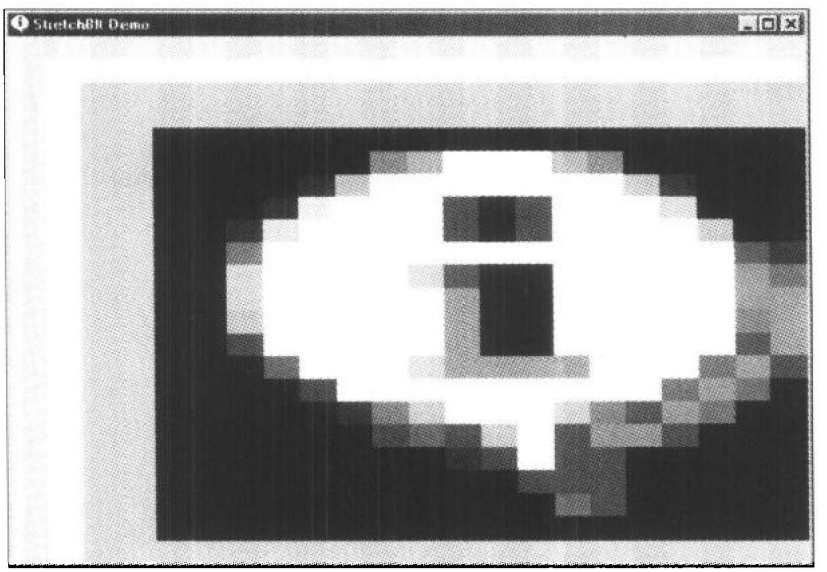


图 14-2　STRETCH 的屏幕显示

BitBlt 和 StretchBlt 函数中所有的坐标与大小都是基于逻辑单位的。但是当您在 BitBlt 函数中有两个不同的设备描述表，而这两个设备描述表引用相同的物理设备却具有不同的映射模式，这时将发生什么情况呢？如果出现这种情况，调用 BitBlt 就显得不明确了：cx 和 cy 参数都是逻辑单位，而它们同样应用于源设备描述表和目标设备描述表中的矩形区。所有的坐标和尺寸必须在实际的位传送之前转换为设备坐标。因为 cx 和 cy 值同时用于源和目标设备描述表，所以此值必须分别转换为设备描述表自己的单位。

当源和目标设备描述表相同，或者两个设备描述表都使用 MM_TEXT 图像模式时，设备单位下的矩形尺寸将在两个设备描述表中相同。然后 Windows 进行像素到像素的转换。不过，当设备单位下的矩形尺寸在两个设备描述表中不同时，Windows 就把此工作转交给更通用的 StretchBlt 函数。

StretchBlt 还允许水平或垂直翻转图像。如果 cxSrc 和 cxDst 标记(转换成设备单位以后)不同,那么 StretchBlt 就创建一个镜像:左右翻转。在 STRETCH 程序中,通过将 xDst 参数改为 cxClient,并将 cxDst 参数改成 - cxClient,您就可以做到这一点。如果 cySrc 和 cyDst 不同,则 StretchBlt 会上下翻转图像。要在 STRETCH 程序中测试这一点,可将 yDst 参数改为 cyClient,并将 cyDst 参数改成 - cyClient。

14.3.3 StretchBlt 模式

StretchBlt 能展示与位图缩放难点相关的一些问题。在扩展一个位图时,StretchBlt 必须复制像素行或列。如果扩展不是原图的整数倍,那么此操作会给图像带来一些失真。

如果目标矩形比源矩形小,那么 StretchBlt 在压缩图像时就必须把两行(或列)或者多行(或列)的像素合并到一行(或列)。完成此操作有 4 种方法,它根据设备描述表中拉伸模式属性来选择其中一种方法。您可使用 SetStretchBltMode 函数来修改属性:

```
SetStretchBltMode (hdc, iMode) ;
```

iMode 可取下列值:

- BLACKONWHITE 或者 STRETCH_ANDSCANS(默认) 如果两个或多个像素组成了一个像素,那么 StretchBlt 对像素执行一个逻辑与操作。这样的结果是只有全部的原始像素是白色时该像素才为白色,其实际意义是黑色像素控制白色像素。这适用于白背景中主要是黑色的单色位图。
- WHITEONBLACK 或 STRETCH_ORSCANS 如果两个或多个像素组成一个像素,则 StretchBlt 执行逻辑或操作。此结果是只有在全部的原始像素都是黑色时才是黑色,也就是说由白色像素决定颜色。这适用于黑色背景中主要是白色的单色位图。
- COLORONCOLOR 或 STRETCH_DELETESCANS StretchBlt 简单地消除像素行或列,而没有任何逻辑组合。这是通常是处理彩色位图的最佳方法。
- HALFTONE 或 STRETCH_HALFTONE Windows 根据组合起来的源颜色来计算目标的平均颜色。这将与中间调调色板联合使用,第十六章将演示这一过程。

Windows 还包括用于获取当前拉伸模式的 GetStretchBltMode 函数。

14.3.4 光栅操作

BITBLT 和 STRETCH 程序简单地将源位图复制给了目标,在此过程中也可能进行了缩放。这是把 SRCCOPY 作为 BitBlt 和 StretchBlt 函数最后一个参数的结果。SRCCOPY 只是能在这些函数中使用的 256 个光栅操作中的一个。让我们先在 STRETCH 程序中做一个实验,然后再系统地研究光栅操作。

用 NOTSRCCOPY 来代替 SRCCOPY 试一试。与它们的名称一样，这个光栅操作在复制位图时反转其颜色。在客户区窗口，所有的颜色被反转：黑色变成白色，白色变成黑色，蓝色变成黄色。再试一下 SRCINVERT，您将得到同样效果。如果试一下 BLACKNESS，正如其名称一样，整个客户区都将变成黑色，而 WHITENESS 则使其变成白色。

现在试一试用下列三条语句来代替 StretchBlt 调用：

```
SelectObject (hdcClient, CreateHatchBrush (HS_DIAGCROSS, RGB (0, 0, 0)));

StretchBlt (hdcClient, 0, 0, cxClient, cyClient,
        hdcWindow, 0, 0, cxSource, cySource, MERGECOPY) ;

DeleteObject (hdcClient, GetStockObject (WHITE_BRUSH)) ;
```

这次，您将看到在图像上叠加一个菱形的画刷，这是为什么？

在前面说过，BitBlt 和 StretchBlt 函数不是简单的位块传送。此函数实际在下面三种图像间执行位操作：

- Source　源位图，拉伸或压缩（如果有必要）到目标矩形的尺寸；
- Destination　在 BitBlt 或 StretchBlt 调用之前的目标矩形；
- Pattern　在目标设备描述表中选择的当前画刷，水平或垂直地重复到目标矩形的尺寸。

结果将复制到目标矩形。

光栅操作与在第五章遇到的绘图模式从概念上是相似的。绘图模式控制图形对象，例如一条线，与目标对象相组合的方式。我们知道有 16 种绘图模式——就是说，所有唯一的结果都是在所绘对象中的 0 和 1 与目标中的 0 和 1 合并时获得的。

使用 BitBlt 和 StretchBlt 的光栅操作包含了三个对象的组合，这将产生 256 种光栅操作。即有 256 种方法来组合源位图、目标位图和图案。有 15 种光栅操作已经命名——其中一些相当含糊——它们在 WINGDI.H 定义，其余的都有数值，在/Platform SDK /Graphics and Multimedia Services/GDI/Raster Operation Codes/Ternary Raster Operations 给出。

有名称的 15 种 ROP 代码显示见下页表 14-4。此表对于理解和使用光栅操作非常重要，因此我们应花点时间来研究。

在该表中，“ROP 代码”列的值是作为 BitBlt 或 StretchBlt 的最后一个参数传递的数；“名称”列中的值在 WINGDI.H 定义。ROP 代码的低位字帮助设备驱动程序传输光栅操作。高位字是 0～255 之间的数字。此数字与第 2 列的图案的位相同，这是在图案、源和显示在顶部的目标之间进行位操作的结果。“布尔操作”列按 C 语言句法显示图案、源和目标的组合方式。

要开始了解此表，最简单的办法是假定您正处理一个单色系统（每像素 1 位），其中

0代表黑色,1代表白色。无论源、目标和图案是什么,BLACKNESS 操作的结果全部为0,因此目标将显示黑色。类似地,WHITENESS 总导致目标呈白色。

表 14-4

图案(P):	11110000			
源(S):	11001100	布尔操作	ROP 代码	名称
目标(D):	10101010			
结果:	00000000	0	0x000042	BLACKNESS
	00010001	~(S ¦ D)	0x1100A6	NOTSRCERASE
	00110011	~S	0x330008	NOTSRCCOPY
	01000100	S & ~D	0x440328	SRCERASE
	01010101	~D	0x550009	DSTINVERT
	01011010	P ∧ D	0x5A0049	PATINVERT
	01100110	S ∧ D	0x660046	SRCINVERT
	10001000	S & D	0x8800C6	SRCAND
	10111011	~S ¦ D	0xBB0226	MERGEPAINT
	11000000	P & S	0xC000CA	MERGECOPY
	11001100	S	0xCC0020	SRCCOPY
	11101110	S ¦ D	0xEE0086	SRCPAINT
	11110000	P	0xF00021	PATCOPY
	11111011	P ¦ ~S ¦ D	0xFB0A09	PATPAINT
	11111111	1	0xFF0062	WHITENESS

现在假定您使用光栅操作 PATCOPY。这导致结果位与图案位相同,而忽略了源和目标位图。换句话说,PATCOPY 简单地将当前图案复制给了目标矩形。

PATPAINT 光栅操作包含一个更复杂的操作。其结果等价于在图案、目标和反转的源之间进行位或操作。当源位图是黑色(0 位)时,其结果总是白色(1 位);当源是白色(1)时,只要图案或目标为白色,则结果就是白色。换句话说,只有源为白色,而图案和目标都是黑色时,结果才是黑色。

彩色显示为每个像素都使用多个位。BitBlt 和 StretchBlt 函数对每个颜色位都分别提供了位操作。例如,如果目标是红色而源为蓝色,则 SRCPAINT 光栅操作把目标变成品红色。注意,操作实际是按视频板内存储的位执行的。这些位所对应的颜色取决于视频卡的调色板的设置。Windows 这样操作,以便光栅操作能达到您预计的结果。不过,如果您修改了调色板(将在第十六章讨论),光栅操作将产生无法预料的结果。

请参见 14.4.10"非矩形位图图像"节的有关内容,可以看到更好地运用光栅操作的应用程序。

14.3.5 图案 Blt

除了 BitBlt 和 StretchBlt 以外，Windows 还包括一个 PatBlt（“pattern block transfer：图案块传送”）函数。这是三个“blt”函数中最简单的。与 BitBlt 和 StretchBlt 不同，它只使用一个目标设备描述表。PatBlt 的语法是：

```
PatBlt (hdc, x, y, cx, cy, dwROP) ;
```

其中 x、y、cx 和 cy 参数是逻辑单位。逻辑点(x,y)指定了矩形的左上角。矩形宽为 cx 单位，高为 cy 单位。这是 PatBlt 将要改变的矩形区域。PatBlt 在画刷与目标设备描述表上执行的逻辑操作由 dwROP 参数决定，此参数是 ROP 代码的子集——也就是说，您可以只使用那些不包括源设备描述表的 ROP 代码。表 14-5 列出了 PatBlt 支持的 16 个光栅操作。

表 14-5

图案(P)：	1 1 0 0	布尔操作	ROP 代码	
目标(D)：	1 0 1 0			名称
结果：	0 0 0 0	0	0x000042	BLACKNESS
	0 0 0 1	~(P ¦ D)	0x0500A9	
	0 0 1 0	~P & D	0x0A0329	
	0 0 1 1	~P	0x0F0001	
	0 1 0 0	P & ~D	0x500325	
	0 1 0 1	~D	0x550009	DSTINVERT
	0 1 1 0	P ∧ D	0x5A0049	PATINVERT
	0 1 1 1	~(P & D)	0x5F00E9	
	1 0 0 0	P & D	0xA000C9	
	1 0 0 1	~(P ∧ D)	0xA50065	
	1 0 1 0	D	0xAA0029	
	1 0 1 1	~P ¦ D	0xAF0229	
	1 1 0 0	P	0xF00021	PATCOPY
	1 1 0 1	P ¦ ~D	0xF50225	
	1 1 1 0	P ¦ D	0xFA0089	
	1 1 1 1	1	0xFF0062	WHITENESS

下面列出了 PatBlt 一些更常见的用途。如果想画一个黑色矩形，您可调用

```
PatBlt (hdc, x, y, cx, cy, BLACKNESS) ;
```

要画一个白色矩形，请用

```
PatBlt (hdc, x, y, cx, cy, WHITENESS) ;
```

函数

```
PatBlt (hdc, x, y, cx, cy, DSTINVERT) ;
```

用于反转矩形的颜色。如果当前设备描述表中选择了 WHITE_BRUSH,那么函数

```
PatBlt (hdc, x, y, cx, cy, PATINVERT) ;
```

也反转矩形。

您可以再次调用 FillRect 函数来用画刷填充一个矩形区域:

```
FillRect (hdc, &rect, hBrush) ;
```

FillRect 函数等价于下列代码:

```
hBrush = SelectObject (hdc, hBrush) ;
PatBlt (hdc, rect.left, rect.top,
          rect.right - rect.left,
          rect.bottom - rect.top, PATCOPY) ;
SelectObject (hdc, hBrush) ;
```

实际上,此代码就是 Windows 用于执行 FillRect 函数的。当您调用

```
InvertRect (hdc, &rect) ;
```

时,Windows 将其转换成函数:

```
PatBlt (hdc, rect.left, rect.top,
          rect.right - rect.left,
          rect.bottom - rect.top, DSTINVERT) ;
```

在介绍 PatBlt 函数的语法时,我说过点(x,y)指出了矩形的左上角,而且此矩形宽度为 cx 单位,高度为 cy 单位。这一点并不完全正确。GDI 画图函数中只有 BitBlt、PatBlt 和 StretchBlt 是根据从一个角测得的逻辑宽度和高度来指定逻辑直角坐标。矩形边框用到的其他所有 GDI 绘图函数都要求根据左上角和右下角的坐标来指定坐标。对于 MM_TEXT 映射模式,上面讲述的 PatBlt 参数就是正确的。然而对于公制的映射模式来说,就不正确。如果使用 cx 和 cy 的正值,那么点(x,y)将是矩形的左下角。如果希望点(x,y)是矩形的左上角,那么 cy 参数必须设为矩形的负高度。

如果想更精确,用 PatBlt 修改颜色的矩形将通过 cx 的绝对值获得逻辑宽度,通过 cy 的绝对值获得逻辑高度。这两个参数可以是负值。由逻辑点(x, y)和(x + cx, y + cy)给定的两个角定义了矩形。矩形的左上角通常属于 PatBlt 修改的区域。右上角则超出了矩形的范围。根据映射模式和 cx、cy 参数的符号,矩形左上角的点应是(x, y)、(x, y + cy)、(x + cx, y),或者(x + cx, y + cy)。

如果设置了 MM_LOENGLISH 映射模式,并且想在客户区左上角的一小块正方形上使用 PatBlt,您可以使用

```
PatBlt (hdc, 0, 0, 100, -100, dwROP) ;
```

或

```
PatBlt (hdc, 0, -100, 100, 100, dwROP) ;
```

或

```
PatBlt (hdc, 100, 0, -100, -100, dwROP) ;
```

或

```
PatBlt (hdc, 100, -100, -100, 100, dwROP) ;
```

给 PatBlt 设定正确参数最容易的方法是将 x 和 y 设为矩形左上角。如果映射模式定义 y 坐标随着向上滚动显示而增加，那么请使用负的 cy 参数。如果映射模式定义 x 坐标向左增加(很少有人用)，则需要使用负的 cx 参数。

14.4 GDI 位图对象

在本章前面已提到过 Windows 从 1.0 版开始就支持 GDI 位图对象。因为在 Windows 3.0 引入了设备无关位图，GDI 位图对象有时也称为设备相关位图或者 DDB。

对程序员来说，现存的两种不同类型的位图从 Windows 3.0 开始就更为混乱。许多有经验的 Windows 程序员都没有准确地理解 DIB 和 DDB 之间的关系。(恐怕本书的 Windows 3.0 版本不能澄清这个问题。)诚然，DIB 和 DDB 在许多方面是相关的：DIB 与 DDB 可以相互转换(尽管转换过程中会丢失一些信息)。然而 DIB 和 DDB 是不可以相互替换的，并且不能简单地选择一种方法来表示同一个可视数据。

如果我们能假定 DIB 废除了 DDB，那会很方便。但并不是这样。DDB 还在 Windows 中扮演着很重要角色，尤其是您关注程序执行时。

14.4.1 创建 DDB

DDB 是 Windows 图形设备接口的图形对象之一(其中还包括画笔、画刷、字体、元文件，以及调色板)。这些图形对象保存在 GDI 模式内部，由应用程序软件的数字句柄引用。您可以将 DDB 句柄存储在一个 HBITMAP("handle to a bitmap：位图句柄")类型的变量中，例如：

```
HBITMAP hBitmap ;
```

然后通过调用某个创建 DDB 的函数来获得句柄，例如 CreateBitmap。这些函数分配并初始化 GDI 内存中的一些内存来存储关于位图的信息，以及实际位图的位信息。应用程序不能直接访问这段内存。位图与设备描述表无关。当程序使用完位图以后，就清除这段内存：

```
DeleteObject (hBitmap) ;
```

如果程序运行时您使用了 DDB,那么程序终止时,您应该执行上面的操作。

CreateBitmap 函数用法如下:

```
hBitmap = CreateBitmap (cx, cy, cPlanes, cBitsPixel, bits) ;
```

前两个参数是位图的宽度和高度(以像素为单位),第 3 个参数是颜色板的数目,第 4 个参数是每像素的位数,第 5 个参数是指向一个位数组的指针,该位与特定颜色格式相匹配。如果您不想用像素位来初始化 DDB,可以将最后一个参数设为 NULL。以后还可以设置像素位。

使用此函数时,Windows 也允许创建您喜欢的特殊类型的 GDI 位图对象。例如,假设您希望位图具有宽 7 个像素、高 9 个像素,以及 5 个颜色板,并且每个像素占 3 位,那么您只需执行下面的操作:

```
hBitmap = CreateBitmap (7, 9, 5, 3, NULL) ;
```

这时 Windows 会返回给您一个有效的位图句柄。

在此函数调用期间,Windows 将保存传递给函数的信息,并为像素位分配内存。粗略的计算是此位图需要 $7 \times 9 \times 5 \times 3$,即 945 位,相当于 118 个字节多一点。

然而,在 Windows 为位图分配内存时,每行像素都具有偶数个字节,即

```
iWidthBytes = 2 * ((cx * cBitsPixel + 15) / 16) ;
```

或者,C 程序员往往把这写成

```
iWidthBytes = (cx * cBitsPixel + 15) & ~15) >> 3 ;
```

因此,为 DDB 分配的内存就是

```
iBitmapBytes = cy * cPlanes * iWidthBytes ;
```

本例中,iWidthBytes 占 4 字节,iBitmapBytes 占 180 字节。

现在,位图就有 5 个颜色板,每像素占 3 个颜色位了吗? 这并不是一个完整的框架,甚至不能把它称作一个学术练习。您会导致 GDI 去分配许多初始内存,并且这些内存有一指定组织,但这并没什么,您不能用这个位图来做任何有用的事。

实际上,您将用两种类型的参数来调用 CreateBitmap:

- cPlanes 和 cBitsPixel 都等于 1(表示单色位图);
- cPlanes 和 cBitsPixel 都等于某个特定设备描述表的值,您可以使用 PLANES 和 BITSPIXEL 索引来从 GetDeviceCaps 函数获得。

对于更现实的情况,您会只在第一种情况下调用 CreateBitmap。对于第二种情况,您可以用 CreateCompatibleBitmap 来简化问题:

```
hBitmap = CreateCompatibleBitmap (hdc, cx, cy) ;
```

此函数创建了一个与设备兼容的位图，此设备的设备描述表句柄由第一个参数给出。CreateCompatibleBitmap 用设备描述表句柄来获得 GetDeviceCaps 信息，然后将此信息传递给 CreateBitmap。除了与实际的设备描述表有相同的内存组织之外，DDB 与设备描述表没有其他联系。

CreateDiscardableBitmap 函数与 CreateCompatibleBitmap 的参数相同，并且功能上等价。在早期的 Windows 版本中，CreateDiscardableBitmap 创建的位图可以在内存不足时由 Windows 将其从内存中清除，然后程序再重建位图数据。

第三个位数创建函数是 CreateBitmapIndirect：

```
hBitmap CreateBitmapIndirect (&bitmap) ;
```

其中 bitmap 是 BITMAP 类型的结构。BITMAP 结构定义如下：

```
typedef struct _tagBITMAP
{
     LONG     bmType ;          // set to 0
     LONG     bmWidth ;         // width in pixels
     LONG     bmHeight ;        // height in pixels
     LONG     bmWidthBytes ;    // width of row in bytes
     WORD     bmPlanes ;        // number of color planes
     WORD     bmBitsPixel ;     // number of bits per pixel
     LPVOID   bmBits ;          // pointer to pixel bits
}
BITMAP, * PBITMAP ;
```

在调用 CreateBitmapIndirect 函数时，您不需要设置 bmWidthBytes 字段。Windows 将为您计算，您也可以将 bmBits 字段设置为 NULL，或者设置为初始化位图时用的像素位地址。

GetObject 函数内也使用 BITMAP 结构，首先定义一个 BITMAP 类型的结构：

```
BITMAP bitmap ;
```

并调用函数如下：

```
GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;
```

Windows 将用位图信息填充 BITMAP 结构的字段，不过，bmBits 字段等于 NULL。

最后应调用 DeleteObject 来清除程序内创建的所有位图。

14.4.2　位图位

用 CreateBitmap 或 CreateBitmapIndirect 来创建设备相关 GDI 位图对象时，您可以给位图像素位指定一个指针。您也可以保持位图未初始化。在创建位图以后，Windows 还提

供两个函数来获得并设置像素位。

要设置像素位,请调用

```
SetBitmapBits (hBitmap, cBytes, &bits) ;
```

GetBitmapBits 函数有相同的语法:

```
GetBitmapBits (hBitmap, cBytes, &bits) ;
```

在这两个函数中,cBytes 指明要复制的字节数,bits 是其大小至少为 cBytes 的缓冲区。

DDB 中的像素位从顶行开始排列。在前面说过,每行的字节数都是偶数。除此之外,没有更多要说明的。如果位图是单色的,也就是说它有 1 个位面,并且每个像素占 1 位,而每个像素或是 1 或是 0。每行最左边的像素是本行第一个字节最重要的位。在本章的后面讲完如何显示单色 DDB 之后,将做一个单色的 DDB。

对于非单色位图,应避免出现在您需要知道像素位含义的位置。例如,假定 Windows 在运行 8 位的 VGA 上,您可以调用 CreateCompatibleBitmap。通过 GetDeviceCaps,您可以确定正处理一个有 1 个颜色板和每像素 8 位的设备。一个字节保存一个像素。但是像素值 0x37 是什么意思呢? 很明显是某种颜色,但到底是什么颜色呢?

像素实际上并不涉及任何固定的颜色,它只是一个值。DDB 没有颜色表。问题的关键在于:当 DDB 显示在屏幕上时,像素的颜色是什么? 它肯定是某种颜色,但具体是什么颜色呢? 显示的像素将是 RGB 颜色,它与在视频板上的调色板查找表里的 0x37 索引值有关。现在使用的设备取决于您。

不过,不要只因为我们不知道像素值的含义就假定非单色 DDB 没用。我们将简要看一下它们的用途。下一章,我们将看到 SetBitmapBits 和 GetBitmapBits 函数是如何被更有用的 SetDIBits 和 GetDIBits 函数所取代。

因此,基本的规则是这样的:不要用 CreateBitmap、CreateBitmapIndirect 或 SetBitmapBits 来设置彩色 DDB 的位,您只能安全地设置单色 DDB 的位。(如果在调用 GetBitmapBits 期间,从其他相同格式的 DDB 中获得位,那么这些规则例外。)

在继续之前,让我再讨论一下 SetBitmapDimensionEx 和 GetBitmapDimensionEx 函数。这些函数允许您设置(和获得)位图的度量尺寸(以 0.1 mm 为单位)。这些信息与位图分辨率一起保存在 GDI 中,但不用于任何操作。它只是您与 DDB 联系的一个度量尺寸标识。

14.4.3 内存设备描述表

我们必须解决的下一个概念是内存设备描述表。您需要用内存设备描述表来处理 GDI 位图对象。

通常,设备描述表指的是某个图形输出设备(例如视频显示器或者打印机),及其设

备驱动程序。内存设备描述表只位于内存中，它不是真正的图形输出设备，但可以说与某个实际设备“兼容”。

要创建一个内存设备描述表，您必须首先有某个实际设备的设备描述表句柄。如果是 hdc，那么您可以像下面那样创建内存设备描述表：

```
hdcMem = CreateCompatibleDC (hdc) ;
```

通常，函数的调用比这更简单。如果将参数设为 NULL，那么 Windows 将创建一个与视频显示器相兼容的内存设备描述表。应用程序创建的任何内存设备描述表最终都通过调用 DeleteDC 来清除。

内存设备描述表有一个与实际光栅设备相同的显示表面。不过，最初此显示表面非常小——单色、1 像素宽和 1 像素高。显示表面就是单独 1 位。

当然，用 1 位的显示表面，您不能做更多的工作，因此下一步就是扩大显示表面。您可以通过将一个 GDI 位图对象选进内存设备描述表来完成这项工作，例如：

```
SelectObject (hdcMem, hBitmap) ;
```

此函数与您将画笔、画刷、字体、区域和调色板选进设备描述表的函数相同。然而，内存设备描述表是您可以选进位图的唯一一种设备描述表类型。（如果需要，您也可以将其他 GDI 对象选进内存设备描述表。）

只有选进内存设备描述表的位图是单色的，或者和与内存设备描述表兼容的设备有相同的色彩结构时，SelectObject 才会起作用。这也是创建特殊的 DDB（例如有 5 个位面，且每像素 3 位）没有用的原因。

SelectObject 调用以后，DDB 就是内存设备描述表的显示表面。处理实际设备描述表的每项操作几乎都可以用于内存设备描述表。例如，如果用 GDI 画图函数在内存设备描述表中画图，那么图像将画在位图上。这是非常有用的。还可以将内存设备描述表作为源，把视频设备描述表作为目标来调用 BitBlt。这就是在显示器上绘制位图的方法。如果把视频设备描述表作为源，把内存设备描述表作为目标，那么调用 BitBlt 可以将屏幕上的一些内容复制给位图。我们将看到这些都是可能的。

14.4.4　加载位图资源

除了各种各样的位图创建函数以外，获得 GDI 位图对象句柄的另一个方法就是调用 LoadBitmap 函数。使用此函数，您不用担心位图格式。在程序中，您只需简单地作为资源来创建位图，这与创建图标或者鼠标光标的方法类似。LoadBitmap 函数的语法与 LoadIcon 和 LoadCursor 相同：

```
hBitmap = LoadBitmap (hInstance, szBitmapName) ;
```

如果想加载系统位图，那么将第一个参数设为 NULL。这些不同的位图是 Windows 可视

界面(例如关闭框和复选标记)的一小部分,它们的标识符以字母 OBM 开始。如果位图与整数标识符有联系,而不是与名称有联系,那么第二个参数就可以使用 MAKEINTRESOURCE 宏。由 LoadBitmap 加载的所有位图最终应用 DeleteObject 清除。

如果位图源是单色的,那么从 LoadBitmap 返回的句柄将指向一个单色的位图对象。如果位图源不是单色,那么从 LoadBitmap 返回的句柄将指向一个 GDI 位图对象,该对象与运行程序的视频显示器有相同的色彩结构。因此,位图始终与视频显示器兼容,并且总是选进与视频显示器相兼容的内存设备描述表。现在不要担心任何色彩变化,采用 LoadBitmap 调用后,这种情况就不会在屏幕上出现。下一章,我们就了解具体的操作。

程序 14-3 所示的 BRICKS1 程序显示了加载一个小的单色位图源的方法。此位图本身不像砖,但当它水平和垂直重复时,就与砖墙相似了。

程序 14-3 BRICKS1 程序

BRICKS1.C

```
/*-------------------
   BRICKS1.C——LoadBitmap Demonstration
              (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Bricks1") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
```

```
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("LoadBitmap Demo"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HBITMAP  hBitmap ;
    static int      cxClient, cyClient, cxSource, cySource ;
    BITMAP          bitmap ;
    HDC             hdc, hdcMem ;
    HINSTANCE       hInstance ;
    int             x, y ;
    PAINTSTRUCT     ps ;

    switch (message)
    {
    case WM_CREATE:
        hInstance = ((LPCREATESTRUCT) lParam) -> hInstance ;

        hBitmap = LoadBitmap (hInstance, TEXT ("Bricks")) ;

        GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;

        cxSource = bitmap.bmWidth ;
        cySource = bitmap.bmHeight ;

        return 0 ;

    case WM_SIZE:
```

```
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;

            for (y = 0 ; y < cyClient ; y += cySource)
            for (x = 0 ; x < cxClient ; x += cxSource)
            {
                BitBlt (hdc, x, y, cxSource, cySource, hdcMem, 0, 0, SRCCOPY) ;
            }

            DeleteDC (hdcMem) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            DeleteObject (hBitmap) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

BRICKS1.RC (摘录)

```
//Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Bitmap
BRICKS                  BITMAP  DISCARDABLE     "Bricks.bmp"
```

BRICKS.BMP

在 Visual C++ Developer Studio 中创建这个位图时，应指明位图的高度和宽度都是 8 个像素，是单色图，名称是“Bricks”。BRICKS1 程序在 WM_CREATE 消息期间加载了位图，并用 GetObject 来确定位图的像素尺寸（以便当位图不是 8 像素见方时，程序仍能继续工作）。以后，BRICKS1 将在 WM_DESTROY 消息中删除此位图。

在 WM_PAINT 消息期间，BRICKS1 创建了一个与显示器兼容的内存设备描述表，并且选进了位图，然后是从内存设备描述表到客户区设备描述表一系列的 BitBlt 函数调用，然后删除内存设备描述表。图 14-3 显示了程序的运行结果。

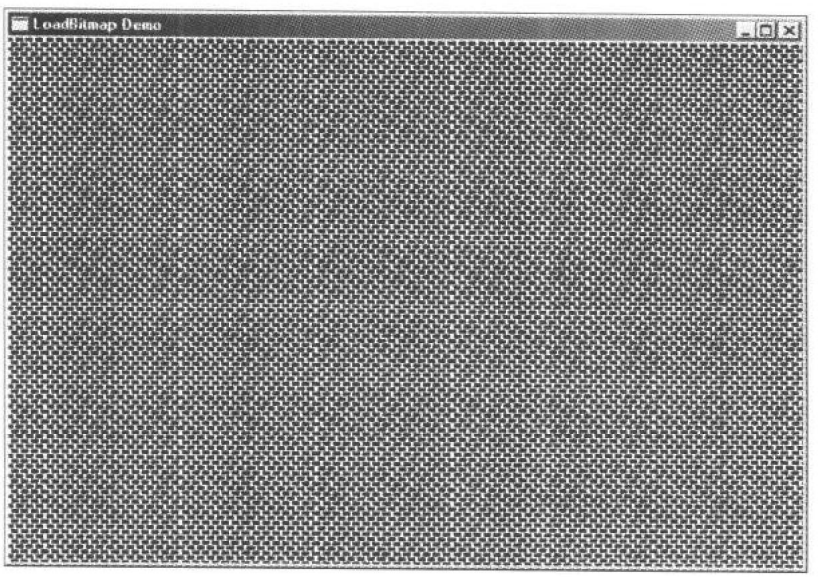


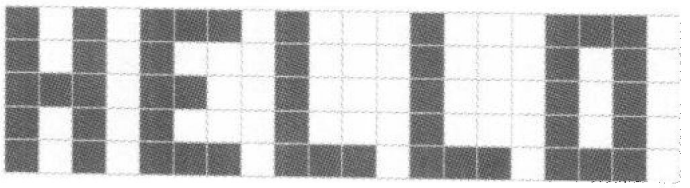
图 14-3　BRICKS1 的屏幕显示

顺便说一下，Developer Studio 创建的 BRICKS.BMP 文件是一个设备无关位图。您可能想在 Developer Studio 内创建一个彩色的 BRICKS.BMP 文件（可自己选定颜色），并且保证一切工作正常。

我们看到 DIB 能转换成与视频显示器兼容的 GID 位图对象。我们将在下一章看到这是如何操作的。

14.4.5　单色位图格式

如果要使用小的单色图像，那么您可以不必将它们作为资源来创建。与彩色位图对象不同，单色位的格式相对简单一些，几乎能够从要创建的图像中直接获得。例如，假定要创建下图所示的位图：



您能写下一系列的位(0 代表黑色,1 代表白色),这些位直接对应于每个网格。从左到右读这些位,您能给每 8 位组分配一个十六进制的字节。如果位图的宽度不是 16 的倍数,在字节的右边用 0 填充,以得到偶数个字节:

```
0 1 0 1 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0 1  =  51 77 10 00
0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1  =  57 77 50 00
0 0 0 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1  =  13 77 50 00
0 1 0 1 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1  =  57 77 50 00
0 1 0 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1  =  51 11 10 00
```

宽为 20 个像素,高为 5 个扫描线,用字节表示宽度是 4。您可以用下面的语句来设置此位图的 BITMAP 结构:

```
static BITMAP bitmap = { 0, 20, 5, 4, 1, 1 } ;
```

并且可以将位保存在 BYTE 数组中:

```
static BYTE bits [ ] = { 0x51, 0x77, 0x10, 0x00,
                         0x57, 0x77, 0x50, 0x00,
                         0x13, 0x77, 0x50, 0x00,
                         0x57, 0x77, 0x50, 0x00,
                         0x51, 0x11, 0x10, 0x00 } ;
```

用 CreateBitmapIndirect 来创建位图需要下面两条语句:

```
bitmap.bmBits = (PSTR) bits ;
hBitmap = CreateBitmapIndirect (&bitmap) ;
```

另一种方法是

```
hBitmap = CreateBitmapIndirect (&bitmap) ;
SetBitmapBits (hBitmap, sizeof bits, bits) ;
```

您也可以用一条语句来创建位图:

```
hBitmap = CreateBitmap (20, 5, 1, 1, bits) ;
```

在程序 14-4 中所示的 BRICKS2 程序利用此技术直接创建了砖位图,而没有使用资源。

程序 14-4　BRICKS2 程序

BRICKS2.C

```
/*-------------------
   BRICKS2.C——CreateBitmap Demonstration
               (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Bricks2") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("CreateBitmap Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
```

```
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BITMAP  bitmap = { 0, 8, 8, 2, 1, 1 } ;
     static BYTE    bits [8][2] = { 0xFF, 0, 0x0C, 0, 0x0C, 0, 0x0C, 0,
                                    0xFF, 0, 0xC0, 0, 0xC0, 0, 0xC0, 0 } ;
     static HBITMAP hBitmap ;
     static int     cxClient, cyClient, cxSource, cySource ;
     HDC            hdc, hdcMem ;
     int            x, y ;
     PAINTSTRUCT    ps ;

     switch (message)
     {
     case WM_CREATE:
          bitmap.bmBits = bits ;
          hBitmap = CreateBitmapIndirect (&bitmap) ;
          cxSource = bitmap.bmWidth ;
          cySource = bitmap.bmHeight ;
          return 0 ;

     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_PAINT:
          hdc = BeginPaint (hwnd, &ps) ;

          hdcMem = CreateCompatibleDC (hdc) ;
          SelectObject (hdcMem, hBitmap) ;
          for (y = 0 ; y < cyClient ; y += cySource)
          for (x = 0 ; x < cxClient ; x += cxSource)
          {
               BitBlt (hdc, x, y, cxSource, cySource, hdcMem, 0, 0, SRCCOPY) ;
          }

          DeleteDC (hdcMem) ;
          EndPaint (hwnd, &ps) ;
```

```
        return 0 ;

    case WM_DESTROY:
        DeleteObject (hBitmap) ;
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

您可以尝试一下与彩色位图相似的对象，例如，如果您的视频显示器运行在 256 色模式下，那么您可以根据 604 页表 14-2 来定义彩色砖的每个像素。不过，当程序运行在其他视频模式下时，此代码不起作用。按设备无关方式处理彩色位图需要使用下章讨论的 DIB。

14.4.6　用位图作画刷

BRICKS 系列的最后一个项目是 BRICKS3，如程序 14-5 所示。初看此程序，您可能会有这种感觉：代码呢？

程序 14-5　BRICKS3 程序

BRICKS3.C

```
/*-----------------------
    BRICKS3.C——CreatePatternBrush Demonstration
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Bricks3") ;
     HBITMAP       hBitmap ;
     HBRUSH        hBrush ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     hBitmap = LoadBitmap (hInstance, TEXT ("Bricks")) ;
     hBrush = CreatePatternBrush (hBitmap) ;
     DeleteObject (hBitmap) ;
```

```
     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = hBrush ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("CreatePatternBrush Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }

     DeleteObject (hBrush) ;
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     switch (message)
     {
     case WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
```

BRICKS3.RC（摘录）

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Bitmap
BRICKS                  BITMAP      DISCARDABLE     "Bricks.bmp"
```

此程序与 BRICKS1 使用同一个 BRICKS.BMP 文件，而且窗口的样子看上去也基本相同。

正如您看到的一样，窗口过程没有更多的内容。BRICKS3 实际上使用砖图案作为窗口类的背景画刷，它在 WNDCLASS 结构的 hbrBackground 字段中定义。

您现在可能猜想 GDI 画刷是一个很小的位图，通常是 8 个像素见方。如果将 LOGBRUSH 结构的 lbStyle 字段设置为 BS_PATTERN，然后调用 CreatePatternBrush 或 CreateBrushIndirect，您就可以用位图来创建画刷。此位图至少是宽高各 8 个像素。如果再大，Windows 98 将只使用位图的左上角作为画刷。而 Windows NT 不受此限制，它会使用整个位图。

请记住，画刷和位图都是 GDI 对象，而且应该在程序终止前删除在程序中创建的所有画刷和位图。如果基于位图创建画刷，那么 Windows 将制作一个在用画刷绘图时使用的位图位的副本。在不影响画刷的情况下调用 CreatePatternBrush（或 CreateBrushIndirect）之后，您可以立即删除位图。类似地，您也可以删除画刷，而不会影响到选进的原始位图。注意，BRICKS3 会在创建画刷之后删除位图，并且在程序终止之前删除画刷。

14.4.7　在位图中绘图

我们已经把位图作为在我们的窗口中进行绘图的一个源了。这要求将位图选进内存设备描述表，并调用 BitBlt 或者 StretchBlt。您也可以使用内存设备描述表句柄作为所有实际调用的 GDI 函数中的第一参数。内存设备描述表的动作与实际的设备描述表的动作相同，除了内存设备描述表的显示表面是一个位图以外。

程序 14-6 所示的 HELLOBIT 程序演示了此项技术。程序在一个小位图上显示了文本串“Hello, world!”，然后执行 BitBlt 或 StretchBlt（根据菜单中的选项），把位图复制到程序客户区。

程序 14-6 HELLOBIT 程序

HELLOBIT.C

```
/* ---------------------
    HELLOBIT.C——Bitmap Demonstration
                (c) Charles Petzold, 1998
   ---------------------*/
# include <windows.h>
# include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    static TCHAR  szAppName [] = TEXT ("HelloBit") ;
    HWND          hwnd ;
    MSG           msg ;
    WNDCLASS      wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = szAppName ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("HelloBit"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
```

```
    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HBITMAP  hBitmap ;
    static HDC      hdcMem ;
    static int      cxBitmap, cyBitmap, cxClient, cyClient, iSize = IDM_BIG ;
    static TCHAR *  szText = TEXT (" Hello, world! ") ;
    HDC             hdc ;
    HMENU           hMenu ;
    int             x, y ;
    PAINTSTRUCT     ps ;
    SIZE            size ;

    switch (message)
    {
    case WM_CREATE:
        hdc = GetDC (hwnd) ;
        hdcMem = CreateCompatibleDC (hdc) ;

        GetTextExtentPoint32 (hdc, szText, lstrlen (szText), &size) ;
        cxBitmap = size.cx ;
        cyBitmap = size.cy ;
        hBitmap = CreateCompatibleBitmap (hdc, cxBitmap, cyBitmap) ;

        ReleaseDC (hwnd, hdc) ;

        SelectObject (hdcMem, hBitmap) ;
        TextOut (hdcMem, 0, 0, szText, lstrlen (szText)) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_COMMAND:
        hMenu = GetMenu (hwnd) ;

        switch (LOWORD (wParam))
        {
        case IDM_BIG:
```

```
            case IDM_SMALL:
                CheckMenuItem (hMenu, iSize, MF_UNCHECKED) ;
                iSize = LOWORD (wParam) ;
                CheckMenuItem (hMenu, iSize, MF_CHECKED) ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                break ;
            }
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            switch (iSize)
            {
            case IDM_BIG:
            StretchBlt (hdc, 0, 0, cxClient, cyClient,
                        hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
            break ;

        case IDM_SMALL:
            for (y = 0 ; y < cyClient ; y += cyBitmap)
            for (x = 0 ; x < cxClient ; x += cxBitmap)
            {
                BitBlt (hdc, x, y, cxBitmap, cyBitmap,
                        hdcMem, 0, 0, SRCCOPY) ;
                }
                break ;
            }

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            DeleteDC (hdcMem) ;
            DeleteObject (hBitmap) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

HELLOBIT.RC (摘录)

```
//Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
```

```
HELLOBIT MENU DISCARDABLE
BEGIN
    POPUP "&Size"
    BEGIN
        MENUITEM "&Big",                IDM_BIG, CHECKED
        MENUITEM "&Small",              IDM_SMALL
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by HelloBit.rc

#define IDM_BIG                         40001
#define IDM_SMALL                       40002
```

程序从调用 GetTextExtentPoint32 来确定文本串的像素尺寸开始。这些尺寸将成为与视频显示兼容的位图尺寸。一旦此位图选进内存设备描述表(也与视频显示兼容),调用 TextOut 将文本显示在位图上。内存设备描述表在程序运行期间保留。在处理 WM_DESTROY 信息期间,HELLOBIT 删除位图和内存设备描述表。

HELLOBIT 有一个菜单选项允许您选择显示位图时的方式,可以按实际尺寸显示(水平和垂直上重复,直至充满整个客户区),也可以放大到整个客户区的大小,如图 14-4所示。正与您所见到的一样,这不是显示大字符的好方法! 它只是小字体的放大版,连锯齿也一同放大了。

图 14-4　HELLOBIT 的屏幕显示

您可能想知道一个程序，例如 HELLOBIT，是否需要处理 WM_DISPLAYCHANGE 消息。只要用户（或者其他应用程序）修改了视频显示大小或者颜色深度，应用程序就接收到此消息，其中颜色深度的改变会导致内存设备描述表和视频设备描述表不兼容。但这并不会发生，因为当视频模式修改后，Windows 自动修改了内存设备描述表的颜色分辨率。选进内存设备描述表的位图也是如此，但它不会导致任何问题。

14.4.8 影子位图

在内存设备描述表（也就是位图）绘图的技术是执行“影子位图（shadow bitmap）”的关键。影子位图包含了窗口客户区中显示的所有内容。这样，对 WM_PAINT 消息的处理就简化到简单的 BitBlt。

影子位图在绘图程序中最有用。程序 14-7 所示的 SKETCH 程序算不上一个严格意义上的绘图程序，但它是一个开始。

程序 14-7 SKETCH 程序

SKETCH.C

```
/*-------------------------
   SKETCH.C——Shadow Bitmap Demonstration
             (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName [] = TEXT ("Sketch") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
```

```
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Sketch"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     if (hwnd == NULL)
     {
          MessageBox (NULL, TEXT ("Not enough memory to create bitmap!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

void GetLargestDisplayMode (int * pcxBitmap, int * pcyBitmap)
{
     DEVMODE devmode ;
     int     iModeNum = 0 ;

     * pcxBitmap = * pcyBitmap = 0 ;

     ZeroMemory (&devmode, sizeof (DEVMODE)) ;
     devmode.dmSize = sizeof (DEVMODE) ;

     while (EnumDisplaySettings (NULL, iModeNum++, &devmode))
     {
          * pcxBitmap = max ( * pcxBitmap, (int) devmode.dmPelsWidth) ;
          * pcyBitmap = max ( * pcyBitmap, (int) devmode.dmPelsHeight) ;
```

```
    }
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BOOL     fLeftButtonDown, fRightButtonDown ;
    static HBITMAP  hBitmap ;
    static HDC      hdcMem ;
    static int      cxBitmap, cyBitmap, cxClient, cyClient, xMouse, yMouse ;
    HDC             hdc ;
    PAINTSTRUCT     ps ;

    switch (message)
    {
    case WM_CREATE:
        GetLargestDisplayMode (&cxBitmap, &cyBitmap) ;

        hdc = GetDC (hwnd) ;
        hBitmap = CreateCompatibleBitmap (hdc, cxBitmap, cyBitmap) ;
        hdcMem = CreateCompatibleDC (hdc) ;
        ReleaseDC (hwnd, hdc) ;

        if (! hBitmap)          // no memory for bitmap
        {
            DeleteDC (hdcMem) ;
            return -1 ;
        }

        SelectObject (hdcMem, hBitmap) ;
        PatBlt (hdcMem, 0, 0, cxBitmap, cyBitmap, WHITENESS) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_LBUTTONDOWN:
        if (!fRightButtonDown)
            SetCapture (hwnd) ;

        xMouse = LOWORD (lParam) ;
        yMouse = HIWORD (lParam) ;
        fLeftButtonDown = TRUE ;
        return 0 ;

    case WM_LBUTTONUP:
        if (fLeftButtonDown)
            SetCapture (NULL) ;
```

```
        fLeftButtonDown = FALSE ;
        return 0 ;

    case WM_RBUTTONDOWN:
        if ( !fLeftButtonDown)
            SetCapture (hwnd) ;

        xMouse = LOWORD (lParam) ;
        yMouse = HIWORD (lParam) ;
        fRightButtonDown = TRUE ;
        return 0 ;

    case WM_RBUTTONUP:
        if (fRightButtonDown)
            SetCapture (NULL) ;

        fRightButtonDown = FALSE ;
        return 0 ;

    case WM_MOUSEMOVE:
        if ( !fLeftButtonDown && ! fRightButtonDown)
            return 0 ;

        hdc = GetDC (hwnd) ;

        SelectObject (hdc,
            GetStockObject (fLeftButtonDown ? BLACK_PEN : WHITE_PEN)) ;

        SelectObject (hdcMem,
            GetStockObject (fLeftButtonDown ? BLACK_PEN : WHITE_PEN)) ;

        MoveToEx (hdc,      xMouse, yMouse, NULL) ;
        MoveToEx (hdcMem,   xMouse, yMouse, NULL) ;

        xMouse = (short) LOWORD (lParam) ;
        yMouse = (short) HIWORD (lParam) ;

        LineTo (hdc,      xMouse, yMouse) ;
        LineTo (hdcMem,   xMouse, yMouse) ;

        ReleaseDC (hwnd, hdc) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
        BitBlt (hdc, 0, 0, cxClient, cyClient, hdcMem, 0, 0, SRCCOPY) ;
```

```
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            DeleteDC (hdcMem) ;
            DeleteObject (hBitmap) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

要想在 SKETCH 中画线，请按下鼠标左键并拖动鼠标；要擦掉（更确切地说，是画白线），请按下鼠标右键并拖动鼠标；要清空整个窗口，请……，哦，必须结束程序，然后重新加载，一切从头再来。图 14-5 显示了 SKETCH 程序向 Apple Macintosh 的早期广告表示自己的敬意。

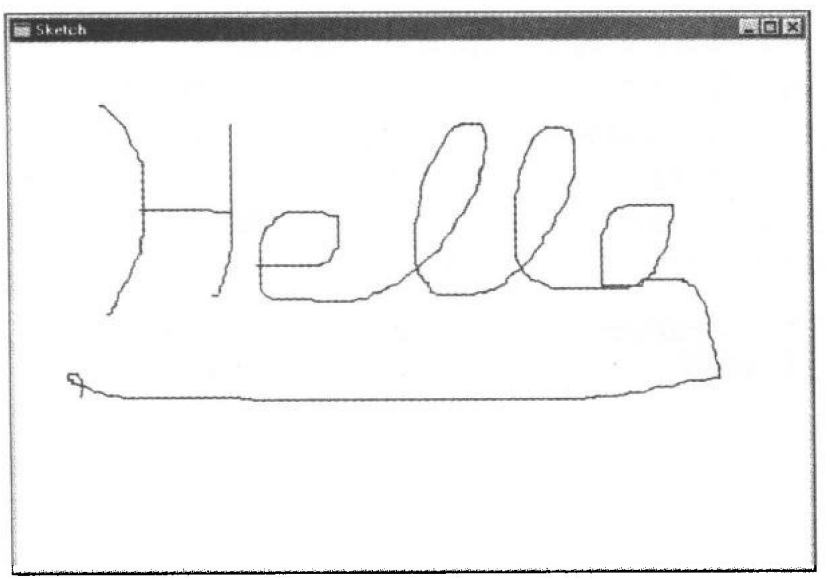


图 14-5　SKETCH 的屏幕显示

此影子位图应多大？在本程序中，它应该大到能包含最大化窗口的整个客户区。这一问题很容易根据 GetSystemMetrics 信息计算得出，但如果用户修改显示设置后再显示，进而扩大了最大化时窗口的尺寸，这时将发生什么呢？SKETCH 程序在 EnumDisplaySettings 函数的帮助下强制解决了此问题。此函数使用 DEVMODE 结构来返回全部有效视频显示模式的信息。第一次调用此函数时，应将 EnumDisplaySettings 的第二参数设为 0，以后每次调用此值都增加。EnumDisplaySettings 返回 FALSE 时完成。

与此同时，SKETCH 将创建一个影子位图，它比当前视频显示模式的表面还多 4 倍，而且需要几兆字节的内存。出于这种原因，SKETCH 将检查是否创建了位图，如果没有创建，就从 WM_CREATE 返回 -1，以表示错误。

在 WM_MOUSEMOVE 消息期间，按下鼠标左键或者右键，并在内存设备描述表和客户区设备描述表中画线时，SKETCH 捕获鼠标。如果画线逻辑更复杂的话，您可能想在一个函数中实现，程序将调用此函数两次——一次为视频设备描述表，一次为内存设备描述表。

下面是一个有趣的实验：使 SKETCH 窗口小于全屏尺寸。按下鼠标左键，将鼠标拖出窗口的右下角。因为 SKETCH 捕获鼠标，所以它继续接收并处理 WM_MOUSEMOVE 消息。现在扩大窗口，您将看到影子位图包含在 SKETCH 窗口外所画的内容。

14.4.9　在菜单中使用位图

您也可以用位图在菜单上显示项目。如果您立即回想起菜单中文件夹、粘贴板和回收站的图片，那么不要再想图片了。应该考虑一下菜单位图对绘图程序有多大用途。想一想使用不同的字体和字号、线宽、阴影图案，以及菜单颜色。

GRAFMENU 是演示图形菜单项目的示例程序。此程序顶级菜单如图 14-6 所示。放大的字母来自于 40 × 16 像素的单色位图文件，该文件用 Visual C++ Developer Studio 创建。从菜单上选择 FONT，将弹出三个选择项——Courier New、Arial 和 Times New Roman。它们是标准的 Windows TrueType 字体，并且每一个都按其对应的字体显示，如图 14-7 所示。这些位图在程序中用内存设备描述表创建。

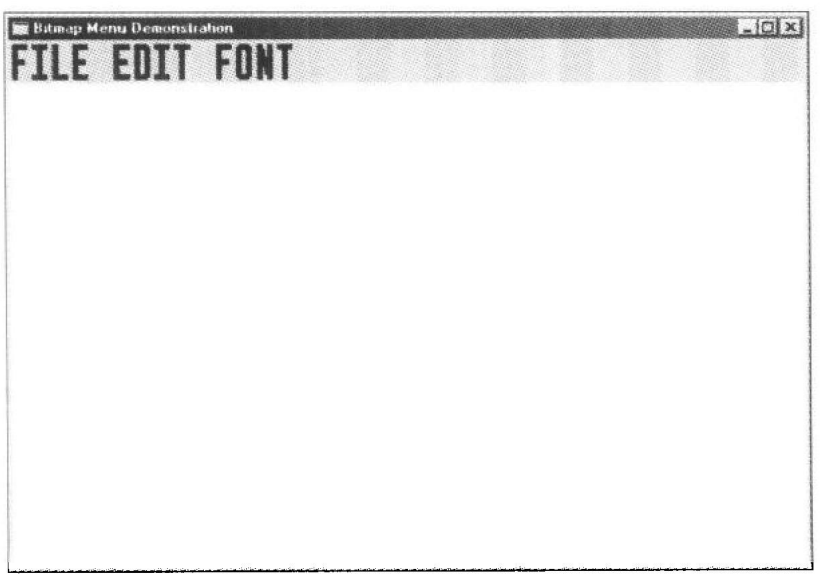


图 14-6　GRAFMENU 程序的顶级菜单

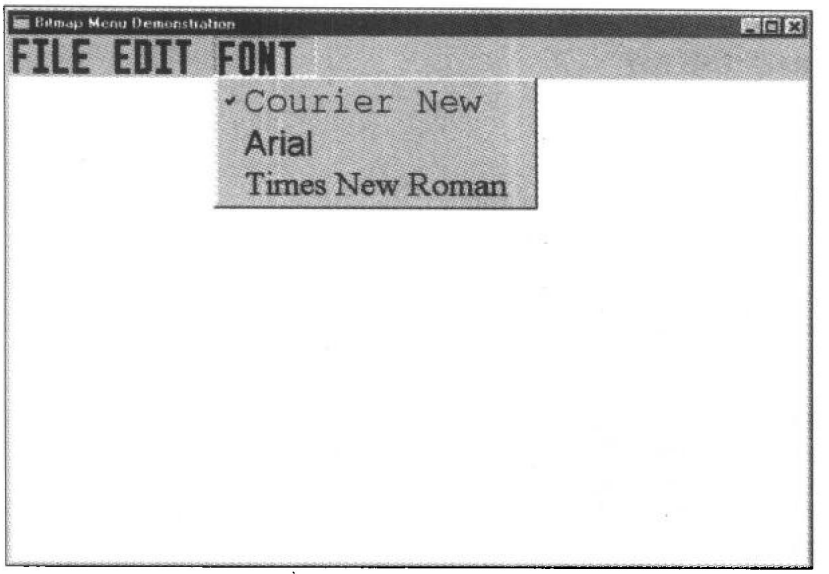


图 14-7　GRAFMENU 程序弹出的“FONT”菜单

最后，在打开系统菜单时，您将有权获得一些“帮助”信息，用“HELP”表示新用户的求助（参见图 14-8）。此 64 像素 × 64 像素的单色位图是在 Developer Studio 中创建的。

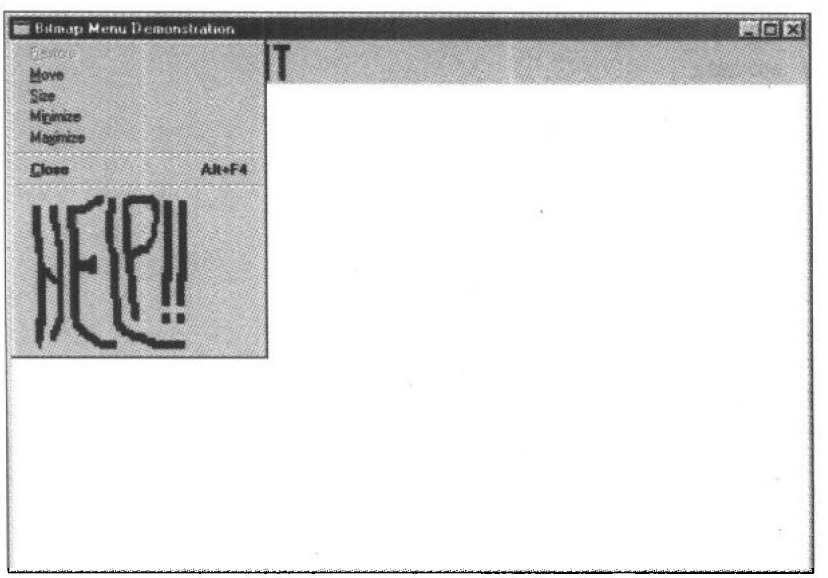


图 14-8　GRAFMENU 程序系统菜单

GRAFMENU 程序，包括 Developer Studio 中创建的 4 个位图，如程序 14-8 所示。

程序 14-8 GRAFMENU 程序

GRAFMENU.C

```
/* ---------------------------
    GRAFMENU.C——Demonstrates Bitmap Menu Items
                (c) Charles Petzold, 1998
   ---------------------------*/

# include <windows.h>
# include "resource.h"

LRESULT  CALLBACK WndProc  (HWND, UINT, WPARAM, LPARAM) ;
void     AddHelpToSys      (HINSTANCE, HWND) ;
HMENU    CreateMyMenu      (HINSTANCE) ;
HBITMAP  StretchBitmap     (HBITMAP) ;
HBITMAP  GetBitmapFont     (int) ;
void     DeleteAllBitmaps  (HWND) ;

TCHAR szAppName[] = TEXT ("GrafMenu") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    HWND        hwnd ;
    MSG         msg ;
    WNDCLASS    wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = NULL ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Bitmap Menu Demonstration"),
```

```
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
     HMENU       hMenu ;
     static int iCurrentFont = IDM_FONT_COUR ;

     switch (iMsg)
     {
     case WM_CREATE:
          AddHelpToSys (((LPCREATESTRUCT) lParam) -> hInstance, hwnd) ;
          hMenu = CreateMyMenu (((LPCREATESTRUCT) lParam) -> hInstance) ;
          SetMenu (hwnd, hMenu) ;
          CheckMenuItem (hMenu, iCurrentFont, MF_CHECKED) ;
          return 0 ;

     case WM_SYSCOMMAND:
          switch (LOWORD (wParam))
          {
          case IDM_HELP:
               MessageBox (hwnd, TEXT ("Help not yet implemented!"),
                           szAppName, MB_OK | MB_ICONEXCLAMATION) ;
               return 0 ;
          }
          break ;

     case WM_COMMAND:
          switch (LOWORD (wParam))
          {
          case IDM_FILE_NEW:
          case IDM_FILE_OPEN:
          case IDM_FILE_SAVE:
          case IDM_FILE_SAVE_AS:
```

```
            case IDM_EDIT_UNDO:
            case IDM_EDIT_CUT:
            case IDM_EDIT_COPY:
            case IDM_EDIT_PASTE:
            case IDM_EDIT_CLEAR:
                MessageBeep (0) ;
                return 0 ;

            case IDM_FONT_COUR:
            case IDM_FONT_ARIAL:
            case IDM_FONT_TIMES:
                hMenu = GetMenu (hwnd) ;
                CheckMenuItem (hMenu, iCurrentFont, MF_UNCHECKED) ;
                iCurrentFont = LOWORD (wParam) ;
                CheckMenuItem (hMenu, iCurrentFont, MF_CHECKED) ;
                return 0 ;
            }
            break ;

        case WM_DESTROY:
            DeleteAllBitmaps (hwnd) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}

/*-----------------------------
    AddHelpToSys: Adds bitmap Help item to system menu
   -----------------------------*/

void AddHelpToSys (HINSTANCE hInstance, HWND hwnd)
{
    HBITMAP  hBitmap ;
    HMENU    hMenu ;

    hMenu = GetSystemMenu (hwnd, FALSE);
    hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapHelp"))) ;
    AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
    AppendMenu (hMenu, MF_BITMAP, IDM_HELP, (PTSTR) (LONG) hBitmap) ;
}

/*-----------------------------
    CreateMyMenu: Assembles menu from components
   -----------------------------*/

HMENU CreateMyMenu (HINSTANCE hInstance)
```

```
{
     HBITMAP  hBitmap ;
     HMENU    hMenu, hMenuPopup ;
     int      i ;

     hMenu = CreateMenu () ;

     hMenuPopup = LoadMenu (hInstance, TEXT ("MenuFile")) ;
     hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapFile"))) ;
     AppendMenu (hMenu, MF_BITMAP | MF_POPUP, (int) hMenuPopup,
                        (PTSTR) (LONG) hBitmap) ;

     hMenuPopup = LoadMenu (hInstance, TEXT ("MenuEdit")) ;
     hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapEdit"))) ;
     AppendMenu (hMenu, MF_BITMAP | MF_POPUP, (int) hMenuPopup,
                        (PTSTR) (LONG) hBitmap) ;

     hMenuPopup = CreateMenu () ;

     for (i = 0 ; i < 3 ; i++)
     {
          hBitmap = GetBitmapFont (i) ;
          AppendMenu (hMenuPopup, MF_BITMAP, IDM_FONT_COUR + i,
                                  (PTSTR) (LONG) hBitmap) ;
     }

     hBitmap = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapFont"))) ;
     AppendMenu (hMenu, MF_BITMAP | MF_POPUP, (int) hMenuPopup,
                        (PTSTR) (LONG) hBitmap) ;
     return hMenu ;
}

/* - - - - - - - - - - - - - - - - - - - - - -
     StretchBitmap: Scales bitmap to display resolution
   - - - - - - - - - - - - - - - - - - - - - - */

HBITMAP StretchBitmap (HBITMAP hBitmap1)
{
     BITMAP     bm1, bm2 ;
     HBITMAP    hBitmap2 ;
     HDC        hdc, hdcMem1, hdcMem2 ;
     int        cxChar, cyChar ;

          // Get the width and height of a system font character

     cxChar = LOWORD (GetDialogBaseUnits ()) ;
     cyChar = HIWORD (GetDialogBaseUnits ()) ;
```

```
        // Create 2 memory DCs compatible with the display

    hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
    hdcMem1 = CreateCompatibleDC (hdc) ;
    hdcMem2 = CreateCompatibleDC (hdc) ;
    DeleteDC (hdc) ;

        // Get the dimensions of the bitmap to be stretched

    GetObject (hBitmap1, sizeof (BITMAP), (PTSTR) &bm1) ;

        // Scale these dimensions based on the system font size

    bm2 = bm1 ;
    bm2.bmWidth         = (cxChar * bm2.bmWidth) / 4 ;
    bm2.bmHeight        = (cyChar * bm2.bmHeight) / 8 ;
    bm2.bmWidthBytes    = ((bm2.bmWidth + 15) / 16) * 2 ;

        // Create a new bitmap of larger size

    hBitmap2 = CreateBitmapIndirect (&bm2) ;

        // Select the bitmaps in the memory DCs and do a StretchBlt

    SelectObject (hdcMem1, hBitmap1) ;
    SelectObject (hdcMem2, hBitmap2) ;
    StretchBlt (hdcMem2, 0, 0, bm2.bmWidth, bm2.bmHeight,
            hdcMem1, 0, 0, bm1.bmWidth, bm1.bmHeight, SRCCOPY) ;

        // Clean up

    DeleteDC (hdcMem1) ;
    DeleteDC (hdcMem2) ;
    DeleteObject (hBitmap1) ;

    return hBitmap2 ;
}

/*-----------------------
    GetBitmapFont: Creates bitmaps with font names
   -----------------------*/

HBITMAP GetBitmapFont (int i)
{
    static TCHAR * szFaceName[3] = { TEXT ("Courier New"), TEXT ("Arial"),
                                     TEXT ("Times New Roman") } ;
    HBITMAP     hBitmap ;
    HDC         hdc, hdcMem ;
    HFONT       hFont ;
```

```
    SIZE            size ;
    TEXTMETRIC      tm ;

    hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
    GetTextMetrics (hdc, &tm) ;

    hdcMem = CreateCompatibleDC (hdc) ;
    hFont = CreateFont (2 * tm.tmHeight, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                        szFaceName[i]) ;

    hFont = (HFONT) SelectObject (hdcMem, hFont) ;
    GetTextExtentPoint32 (hdcMem, szFaceName[i],
                          lstrlen (szFaceName[i]), &size);

    hBitmap = CreateBitmap (size.cx, size.cy, 1, 1, NULL) ;
    SelectObject (hdcMem, hBitmap) ;

    TextOut (hdcMem, 0, 0, szFaceName[i], lstrlen (szFaceName[i])) ;

    DeleteObject (SelectObject (hdcMem, hFont)) ;
    DeleteDC (hdcMem) ;
    DeleteDC (hdc) ;

    return hBitmap ;
}

/* ----------------------------
    DeleteAllBitmaps: Deletes all the bitmaps in the menu
   ---------------------------- */

void DeleteAllBitmaps (HWND hwnd)
{
    HMENU           hMenu ;
    int             i ;
    MENUITEMINFO    mii = { sizeof (MENUITEMINFO), MIIM_SUBMENU | MIIM_TYPE } ;
        // Delete Help bitmap on system menu

    hMenu = GetSystemMenu (hwnd, FALSE);
    GetMenuItemInfo (hMenu, IDM_HELP, FALSE, &mii) ;
    DeleteObject ((HBITMAP) mii.dwTypeData) ;

        // Delete top-level menu bitmaps

    hMenu = GetMenu (hwnd) ;

    for (i = 0 ; i < 3 ; i++ )
    {
        GetMenuItemInfo (hMenu, i, TRUE, &mii) ;
```

```
            DeleteObject ((HBITMAP) mii.dwTypeData) ;
        }

            // Delete bitmap items on Font menu

        hMenu = mii.hSubMenu ;;

        for (i = 0 ; i < 3 ; i++ )
        {
            GetMenuItemInfo (hMenu, i, TRUE, &mii) ;
            DeleteObject ((HBITMAP) mii.dwTypeData) ;
        }
}
```

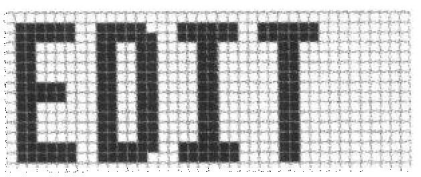
GRAFMENU.RC (摘录)

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
MENUFILE MENU DISCARDABLE
BEGIN
    MENUITEM "&New ",                   IDM_FILE_NEW
    MENUITEM "&Open...",                IDM_FILE_OPEN
    MENUITEM "&Save ",                  IDM_FILE_SAVE
    MENUITEM "Save &As...",             IDM_FILE_SAVE_AS
END

MENUEDIT MENU DISCARDABLE
BEGIN
    MENUITEM "&Undo ",                  IDM_EDIT_UNDO
    MENUITEM SEPARATOR
    MENUITEM "Cu&t ",                   IDM_EDIT_CUT
    MENUITEM "&Copy ",                  IDM_EDIT_COPY
    MENUITEM "&Paste ",                 IDM_EDIT_PASTE
    MENUITEM "De&lete ",                IDM_EDIT_CLEAR
END

//////////////////////////////////////////////////////////////////////////////
// Bitmap
BITMAPFONT      BITMAP  DISCARDABLE     "Fontlabl.bmp"
BITMAPHELP      BITMAP  DISCARDABLE     "Bighelp.bmp"
BITMAPEDIT      BITMAP  DISCARDABLE     "Editlabl.bmp"
BITMAPFILE      BITMAP  DISCARDABLE     "Filelabl.bmp"
```

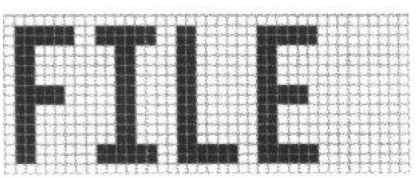
RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by GrafMenu.rc

# define IDM_FONT_COUR          101
# define IDM_FONT_ARIAL         102
# define IDM_FONT_TIMES         103
# define IDM_HELP               104
# define IDM_EDIT_UNDO          40005
# define IDM_EDIT_CUT           40006
# define IDM_EDIT_COPY          40007
# define IDM_EDIT_PASTE         40008
# define IDM_EDIT_CLEAR         40009
# define IDM_FILE_NEW           40010
# define IDM_FILE_OPEN          40011
# define IDM_FILE_SAVE          40012
# define IDM_FILE_SAVE_AS       40013
```

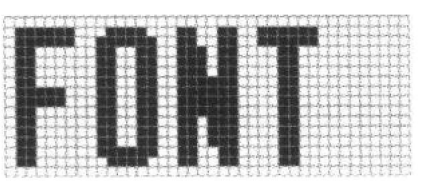
EDITLABL.BMP



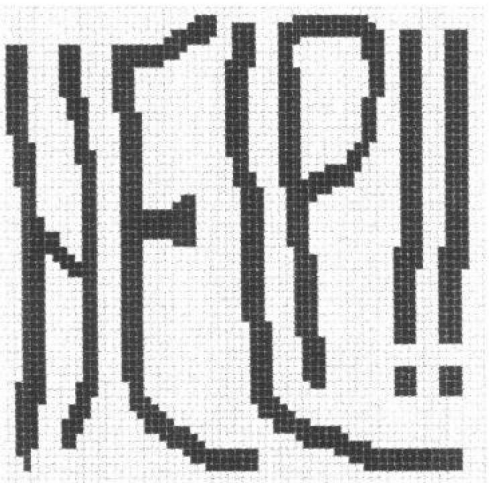
FILELABL.BMP



FONTLABL.BMP



BIGHELP.BMP



要将位图插入菜单，可以利用 AppendMenu 或 InsertMenu。位图有两个来源：可以用 Visual C++ Developer Studio 创建位图，包括资源脚本中的位图文件，并在程序使用 LoadBitmap 时，将位图资源加载到内存，然后调用 AppendMenu 或 InsertMenu，将位图添加到菜单上。但是用这种方法会有一些问题：位图不一定适用于所有视频类型的分辨率和纵横比；有时您需要拉伸加载的位图以解决此问题。另一种方法是：在程序内部创建位图，并将它选进内存设备描述表，在其中绘画，然后再添加到菜单中。

GRAFMENU 中的 GetBitmapFont 函数的参数为 0、1 或 2，返回一个位图句柄。此位图包含字符串 Courier New、Arial 或 Times New Roman，而且字体是各自对应的字体，大小是正常系统字体的两倍。让我们看看 GetBitmapFont 是怎么做的。（下面的代码与 GRAFMENU.C 文件中的有些不同。为了清楚起见，我用与 Arial 字体相应的值代替引用 szFaceName 数组。）

第一步是用 TEXTMETRIC 结构来确定当前系统字体的大小，并创建一个与当前屏幕兼容的内存设备描述表：

```
hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
GetTextMetrics (hdc, &tm) ;
hdcMem = CreateCompatibleDC (hdc) ;
```

CreateFont 函数创建了一种逻辑字体，该字体高是系统字体的两倍，而且逻辑名称为“Arial”：

```
hFont = CreateFont (2 * tm.tmHeight, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                    TEXT ("Arial")) ;
```

从内存设备描述表中选择字体，然后保存默认字体句柄：

```
hFont = (HFONT) SelectObject (hdcMem, hFont) ;
```

现在，当我们向内存设备描述表写一些文本时，Windows 就会使用选进设备描述表的 TrueType Arial 字体。

但这个内存设备描述表最初只有一个单像素单色设备表面。我们必须创建一个足够大的位图，以容纳所要显示的文本。通过 GetTextExtentPoint32 函数，可以获取文本的大小，而用 CreateBitmap 可以根据这些尺寸来创建位图：

```
GetTextExtentPoint32 (hdcMem, TEXT ("Arial"), 5, &size) ;
hBitmap = CreateBitmap (size.cx, size.cy, 1, 1, NULL) ;
SelectObject (hdcMem, hBitmap) ;
```

现在这个设备描述表是一个单色的显示表面，大小也是严格的文本尺寸。现在要做的就是书写文本：

```
TextOut (hdcMem, 0, 0, TEXT ("Arial"), 5) ;
```

除了清除，所有的工作都完成了。要清除，我们可以用 SelectObject 将系统字体（带有句柄 hFont）重新选进设备描述表，然后删除 SelectObject 返回的前一个字体句柄，也就是 Arial 字体句柄：

```
DeleteObject (SelectObject (hdcMem, hFont)) ;
```

现在可以删除两个设备描述表：

```
DeleteDC (hdcMem) ;
DeleteDC (hdc) ;
```

这样，我们就获得了一个位图，该位图上有 Arial 字体的字符串 Arial。

当我们需要缩放字体以适应不同显示分辨率或纵横比时，内存设备描述表也能解决问题。在 GRAFMENU 程序中，我创建了 4 个位图，这些位图只适用于系统字体高 8 像素和宽 4 像素的显示。对于其他尺寸的系统字体，只能缩放位图。GRAFMENU 中的 StretchBitmap 函数完成此功能。

第一步是获得显示的设备描述表，然后获取系统字体的文本规格，接下来创建两个内存设备描述表：

```
hdc = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
GetTextMetrics (hdc, &tm) ;
hdcMem1 = CreateCompatibleDC (hdc) ;
hdcMem2 = CreateCompatibleDC (hdc) ;
DeleteDC (hdc) ;
```

传递给函数的位图句柄是 hBitmap1。程序能用 GetObject 获得位图的大小：

```
GetObject (hBitmap1, sizeof (BITMAP), (PSTR) &bm1) ;
```

此操作将尺寸复制到 BITMAP 类型的结构 bm1 中。结构 bm2 等于结构 bm1,然后根据系统字体大小来修改某些字段:

```
bm2 = bm1 ;
bm2.bmWidth      = (tm.tmAveCharWidth * bm2.bmWidth) / 4 ;
bm2.bmHeight     = (tm.tmHeight       * bm2.bmHeight) / 8 ;
bm2.bmWidthBytes = ((bm2.bmWidth + 15) / 16) * 2 ;
```

下面就根据这个新的尺寸创建一个新的位图,句柄是 hBitmap2:

```
hBitmap2 = CreateBitmapIndirect (&bm2) ;
```

然后将这两个位图选进两个内存设备描述表中:

```
SelectObject (hdcMem1, hBitmap1) ;
SelectObject (hdcMem2, hBitmap2) ;
```

我们想把第一个位图复制给第二个位图,并在此过程中进行拉伸。这需要 StretchBlt 调用:

```
StretchBlt (hdcMem2, 0, 0, bm2.bmWidth, bm2.bmHeight,
        hdcMem1, 0, 0, bm1.bmWidth, bm1.bmHeight, SRCCOPY) ;
```

现在第二幅图适当地缩放了,我们可将其用到菜单中。剩下的清除工作很简单:

```
DeleteDC (hdcMem1) ;
DeleteDC (hdcMem2) ;
DeleteObject (hBitmap1) ;
```

在构造菜单时,GRAFMENU 中的 CreateMyMenu 函数调用 StretchBitmap 和 GetBitmapFont 函数。GRAFMENU 在资源文件中定义了两个菜单,在选择 File 和 Edit 选项时会弹出这两个菜单。函数开始先获取一个空菜单的句柄:

```
hMenu = CreateMenu () ;
```

从资源文件加载 File 的弹出菜单(包括 4 个选项:New、Open、Save 和 Save as):

```
hMenuPopup = LoadMenu (hInstance, TEXT ("MenuFile")) ;
```

从资源文件还加载了包含"FILE"的位图,并用 StretchBitmap 进行了拉伸:

```
hBitmapFile = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapFile"))) ;
```

位图句柄和弹出菜单句柄都是 AppendMenu 调用的参数:

```
AppendMenu (hMenu, MF_BITMAP | MF_POPUP, hMenuPopup, (PTSTR) (LONG) hBitmapFile) ;
```

对 Edit 菜单有如下相同过程:

```
hMenuPopup = LoadMenu (hInstance, TEXT ("MenuEdit")) ;
hBitmapEdit = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapEdit"))) ;
AppendMenu (hMenu, MF_BITMAP | MF_POPUP, hMenuPopup, (PTSTR) (LONG) hBitmapEdit) ;
```

调用 GetBitmapFont 函数可以构造这三种不同字体的弹出式菜单：

```
hMenuPopup = CreateMenu () ;
for (i = 0 ; i < 3 ; i++)
{
     hBitmapPopFont [i] = GetBitmapFont (i) ;
     AppendMenu (hMenuPopup, MF_BITMAP, IDM_FONT_COUR + i,
                              (PTSTR) (LONG) hMenuPopupFont [i]) ;
}
```

然后将弹出式菜单添加到菜单中：

```
hBitmapFont = StretchBitmap (LoadBitmap (hInstance, "BitmapFont")) ;
AppendMenu (hMenu, MF_BITMAP | MF_POPUP, hMenuPopup, (PTSTR) (LONG) hBitmapFont) ;
```

WndProc 通过调用 SetMenu，完成了窗口菜单的创建工作。

GRAFMENU 还改变了 AddHelpToSys 函数中的系统菜单。此函数首先获得一个系统菜单句柄：

```
hMenu = GetSystemMenu (hwnd, FALSE) ;
```

这将加载“HELP”位图，并将其放大到适当尺寸：

```
hBitmapHelp = StretchBitmap (LoadBitmap (hInstance, TEXT ("BitmapHelp"))) ;
```

这将给系统菜单添加一条分隔线和一个拉伸的位图：

```
AppendMenu (hMenu, MF_SEPARATOR, 0, NULL) ;
AppendMenu (hMenu, MF_BITMAP, IDM_HELP, (PTSTR)(LONG) hBitmapHelp) ;
```

GRAFMENU 在退出之前调用一个函数来清除并删除所有位图。

下面是在菜单中使用位图的一些注意事项。

在顶级菜单中，Windows 调整菜单栏的高度，以适应最高的位图。其他位图（或字符串）根据菜单栏的顶端对齐。如果在顶级菜单中使用了位图，那么从使用常量 SM_CYMENU 的 GetSystemMetrics 得到的菜单栏大小将不再有效。

运行 GRAFMENU 期间可以看到：在弹出式菜单中，您可以使用带有位图菜单项的复选标记，但复选标记是正常尺寸。如果不满意，您可以创建一个自定义的复选标记，并使用 SetMenuItemBitmaps。

在菜单中使用非文本（或者使用非系统字体的文本）的另一种方法是“拥有者自绘”菜单。

菜单的键盘接口是另一个问题。当菜单含有文本时，Windows 会自动添加键盘接

口。要选择一个菜单项，可以使用 Alt 键与字符串中的一个字母的组合键。而一旦在菜单中放置了位图，就会删除键盘接口。即使位图表达了一定的含义，但 Windows 并不知道。

目前我们可以使用 WM_MENUCHAR 消息。当您按下 Alt 键和与菜单项不相符的一个字符键的组合键时，Windows 将向您的窗口过程发送一个 WM_MENUCHAR 消息。GRAFMENU 需要截取 WM_MENUCHAR 消息并检查 wParam 的值(即按键的 ASCII 码)。如果这个值对应一个菜单项，那么向 Windows 返回双字：其中高位字为 2，低位字是与该键相关的菜单项索引值，然后由 Windows 处理余下的事。

14.4.10　非矩形位图图像

位图都是矩形，但不需要都显示成矩形。例如，假定有一个矩形位图图像，但却想将它显示成椭圆形。

首先，这听起来很简单。您只需将图像调入 Visual C++ Developer Studio 或者 Windows 的“画图”程序(或者更高级的应用程序)中，然后用白色的画笔将图像四周画上白色。这时将获得一幅椭圆形的图像，而椭圆的外面就成了白色。但是只有当背景色为白色时此位图才能正确显示，如果在其他背景色上显示，您就会发现椭圆形的图像和背景之间有一个白色的矩形。这种效果并不好。

有一种非常通用的技术可解决此类问题。这种技术包括“掩码(mask)”位图和一些光栅操作。掩码是一种单色位图，它与要显示的矩形位图图像尺寸相同。每个掩码的像素都对应位图图像的一个像素。掩码像素为 1(白色)，对应着原始位图像素显示；掩码像素为 0(黑色)，则表示保留目标背景色。(或者掩码位图与此相反，这根据您使用的光栅操作而有一些相应的变化。)

让我们看看 BITMASK 程序是如何运用这一技术的。如程序 14-9 所示。

程序 14-9　BITMASK 程序

BITMASK.C

```
/*-----------------------
   BITMASK.C——Bitmap Masking Demonstration
              (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
```

```
                PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName [] = TEXT ("BitMask") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (LTGRAY_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Bitmap Masking Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP     hBitmapImag, hBitmapMask ;
     static HINSTANCE   hInstance ;
     static int         cxClient, cyClient, cxBitmap, cyBitmap ;
```

```
BITMAP          bitmap ;
HDC             hdc, hdcMemImag, hdcMemMask ;
int             x, y ;
PAINTSTRUCT     ps ;

switch (message)
{
case WM_CREATE:
    hInstance = ((LPCREATESTRUCT) lParam) -> hInstance ;

        // Load the original image and get its size

    hBitmapImag = LoadBitmap (hInstance, TEXT ("Matthew")) ;
    GetObject (hBitmapImag, sizeof (BITMAP), &bitmap) ;
    cxBitmap = bitmap.bmWidth ;
    cyBitmap = bitmap.bmHeight ;

        // Select the original image into a memory DC
    hdcMemImag = CreateCompatibleDC (NULL) ;
    SelectObject (hdcMemImag, hBitmapImag) ;

        // Create the monochrome mask bitmap and memory DC

    hBitmapMask = CreateBitmap (cxBitmap, cyBitmap, 1, 1, NULL) ;
    hdcMemMask = CreateCompatibleDC (NULL) ;
    SelectObject (hdcMemMask, hBitmapMask) ;

        // Color the mask bitmap black with a white ellipse

    SelectObject (hdcMemMask, GetStockObject (BLACK_BRUSH)) ;
    Rectangle (hdcMemMask, 0, 0, cxBitmap, cyBitmap) ;
    SelectObject (hdcMemMask, GetStockObject (WHITE_BRUSH)) ;
    Ellipse (hdcMemMask, 0, 0, cxBitmap, cyBitmap) ;

        // Mask the original image

    BitBlt (hdcMemImag, 0, 0, cxBitmap, cyBitmap,
          hdcMemMask, 0, 0, SRCAND) ;

    DeleteDC (hdcMemImag) ;
    DeleteDC (hdcMemMask) ;
    return 0 ;

case WM_SIZE:
    cxClient = LOWORD (lParam) ;
    cyClient = HIWORD (lParam) ;
    return 0 ;
```

```
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

            // Select bitmaps into memory DCs

        hdcMemImag = CreateCompatibleDC (hdc) ;
        SelectObject (hdcMemImag, hBitmapImag) ;

        hdcMemMask = CreateCompatibleDC (hdc) ;
        SelectObject (hdcMemMask, hBitmapMask) ;

            // Center image

        x = (cxClient - cxBitmap) / 2 ;
        y = (cyClient - cyBitmap) / 2 ;

            // Do the bitblts

        BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemMask, 0, 0, 0x220326) ;
        BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemImag, 0, 0, SRCPAINT) ;

        DeleteDC (hdcMemImag) ;
        DeleteDC (hdcMemMask) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        DeleteObject (hBitmapImag) ;
        DeleteObject (hBitmapMask) ;
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

BITMASK.RC

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Bitmap
MATTHEW                 BITMAP      DISCARDABLE     "matthew.bmp"
```

资源文件中的 MATTHEW.BMP 文件是我侄子的一幅黑白数字照片，宽 200 像素，高 320 像素，每像素 8 位。不过，BITMASK 允许您使用其他任何图像文件。

注意，BITMASK 将窗口背景设为亮灰色。这样就确保能正确地屏蔽位图，而不只是将其涂成白色。

下面让我们看一下 WM_CREATE 的处理过程：BITMASK 用 LoadBitmap 函数获得 hBitmapImag 变量中原始图像的句柄，用 GetObject 函数可获取位图的宽度和高度，然后将位图句柄选进句柄为 hdcMemImag 的内存设备描述表中。

下一步程序创建一个与原位图大小相同的单色位图，其句柄保存在 hBitmapMask，并选进句柄为 hdcMemMask 的内存设备描述表中。在内存设备描述表中，使用 GDI 函数，掩码位图就涂成了黑色背景和一个白色的椭圆：

```
SelectObject (hdcMemMask, GetStockObject (BLACK_BRUSH)) ;
Rectangle (hdcMemMask, 0, 0, cxBitmap, cyBitmap) ;
SelectObject (hdcMemMask, GetStockObject (WHITE_BRUSH)) ;
Ellipse (hdcMemMask, 0, 0, cxBitmap, cyBitmap) ;
```

因为这是一个单色的位图，所以黑色区域的位是 0，而白色区域的位是 1。

然后，BitBlt 调用就按此掩码修改原图像：

```
BitBlt (hdcMemImag, 0, 0, cxBitmap, cyBitmap,
        hdcMemMask, 0, 0, SRCAND) ;
```

SRCAND 光栅操作在源位（掩码位图）和目标位（原图像）之间执行了按位与操作。只要掩码位图是白色，就显示目标；只要掩码是黑色，则目标也是黑色。现在原图像中就形成了一个黑色包围的椭圆区域。

现在让我们看一下 WM_PAINT 处理过程。此过程同时改变了选进内存设备描述表中的图像位图和掩码位图。两次 BitBlt 调用完成了这个魔术。第一次在窗口上执行掩码位图的 BitBlt：

```
BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemMask, 0, 0, 0x220326) ;
```

这里使用了一个没有名称的光栅操作。逻辑操作是 D & ~S。请注意源——掩码位图——是黑色（0 位）包围的一个白色（1 位）椭圆。光栅操作首先将源反色，也就是改成白色包围的黑色椭圆，然后位操作在这个已转换的源和目标（即窗口表面）之间执行位与操作。当目标和 1 位“与”时保持不变；与 0 位“与”时，目标将变黑。因此，BitBlt 操作将在窗口上画一个黑色的椭圆。

第二次的 BitBlt 调用则在窗口中绘制图像位图：

```
BitBlt (hdc, x, y, cxBitmap, cyBitmap, hdcMemImag, 0, 0, SRCPAINT) ;
```

光栅操作在源和目标之间执行位“或”操作。由于源位图的外面是黑色，因此保持目标不变；而在椭圆区域内，目标是黑色，因此图像就原封不动地复制过来。运行结果如图 14-9所示。

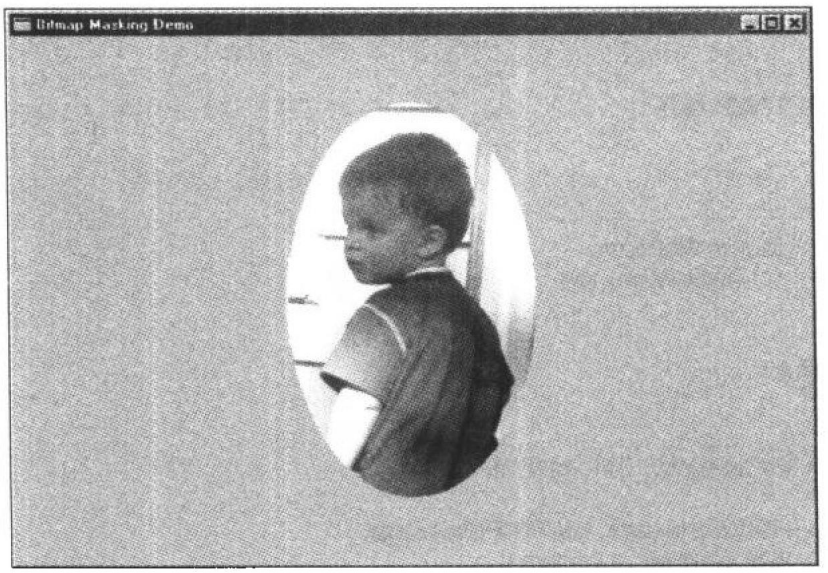


图 14-9　BITMASK 的屏幕显示

有几点需要注意：

有时需要一个很复杂的掩码——例如，抹去原始图像的整个背景。您将需要在绘图程序中手工创建，然后将其保存到文件中。

如果正在为 Windows NT 编写类似的应用程序，那么您可以使用与 MASKBIT 程序类似的 MaskBlt 函数，减少函数调用。Windows NT 还包括另一个类似 BitBlt 的函数，该函数不支持 Windows 98。此函数是 PlgBlt（“平行四边形位块移动：parallelogram blt”）。这个函数可以对图像进行旋转，或者倾斜位图图像。

最后，如果在您的机器上运行 BITMASK 程序，也许只能看见黑色、白色和一些灰色的阴影，这是因为您运行的显示模式是 16 色或 256 色。对于 16 色模式，显示效果无法改进，但在 256 色模式下可以改变调色板，以显示灰度。您将在第十六章学会如何设置调色板。

14.4.11　简单的动画

小的位图显示起来非常快，因此可以将位图和 Windows 计时器结合使用来完成一些基本的动画。

现在开始讨论这个弹球程序。

BOUNCE 程序，如程序 14-10 所示，构造了一个在窗口客户区弹来弹去的小球。该程序利用计时器来控制小球的行进速度。小球本身是一幅位图，程序首先通过创建位图来创建小球，将其选进内存设备描述表，然后调用一些简单的 GDI 函数。程序用 Bit-

Blt 从一个内存设备描述表将这个位图小球画到显示器中。

程序 14-10　BOUNCE 程序

BOUNCE.C

```
/*-------------------
   BOUNCE.C——Bouncing Ball Program
              (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>
#define ID_TIMER    1

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Bounce") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Bouncing Ball"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
```

```
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP  hBitmap ;
     static int      cxClient, cyClient, xCenter, yCenter, cxTotal, cyTotal,
                     cxRadius, cyRadius, cxMove, cyMove, xPixel, yPixel ;
     HBRUSH          hBrush ;
     HDC             hdc, hdcMem ;
     int             iScale ;

     switch (iMsg)
     {
     case WM_CREATE:
          hdc = GetDC (hwnd) ;
          xPixel = GetDeviceCaps (hdc, ASPECTX) ;
          yPixel = GetDeviceCaps (hdc, ASPECTY) ;
          ReleaseDC (hwnd, hdc) ;

          SetTimer (hwnd, ID_TIMER, 50, NULL) ;
          return 0 ;

     case WM_SIZE:
          xCenter = (cxClient = LOWORD (lParam)) / 2 ;
          yCenter = (cyClient = HIWORD (lParam)) / 2 ;

          iScale = min (cxClient * xPixel, cyClient * yPixel) / 16 ;

          cxRadius = iScale / xPixel ;
          cyRadius = iScale / yPixel ;

          cxMove = max (1, cxRadius / 2) ;
          cyMove = max (1, cyRadius / 2) ;

          cxTotal = 2 * (cxRadius + cxMove) ;
          cyTotal = 2 * (cyRadius + cyMove) ;

          if (hBitmap)
               DeleteObject (hBitmap) ;
```

```
            hdc = GetDC (hwnd) ;
            hdcMem = CreateCompatibleDC (hdc) ;
            hBitmap = CreateCompatibleBitmap (hdc, cxTotal, cyTotal) ;
            ReleaseDC (hwnd, hdc) ;

            SelectObject (hdcMem, hBitmap) ;
            Rectangle (hdcMem, -1, -1, cxTotal + 1, cyTotal + 1) ;

            hBrush = CreateHatchBrush (HS_DIAGCROSS, 0L) ;
            SelectObject (hdcMem, hBrush) ;
            SetBkColor (hdcMem, RGB (255, 0, 255)) ;
            Ellipse (hdcMem, cxMove, cyMove, cxTotal - cxMove, cyTotal - cyMove) ;
            DeleteDC (hdcMem) ;
            DeleteObject (hBrush) ;
            return 0 ;

        case WM_TIMER:
            if (!hBitmap)
                break ;

            hdc = GetDC (hwnd) ;
            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;

            BitBlt (hdc, xCenter - cxTotal / 2,
                         yCenter - cyTotal / 2, cxTotal, cyTotal,
                    hdcMem, 0, 0, SRCCOPY) ;

            ReleaseDC (hwnd, hdc) ;
            DeleteDC (hdcMem) ;

            xCenter += cxMove ;
            yCenter += cyMove ;

            if ((xCenter + cxRadius >= cxClient) || (xCenter - cxRadius <= 0))
                cxMove = -cxMove ;

            if ((yCenter + cyRadius >= cyClient) || (yCenter - cyRadius <= 0))
                cyMove = -cyMove ;

            return 0 ;

        case WM_DESTROY:
            if (hBitmap)
                DeleteObject (hBitmap) ;

            KillTimer (hwnd, ID_TIMER) ;
            PostQuitMessage (0) ;
```

```
            return 0 ;
        }
    return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
```

BOUNCE 无论在什么时候接收到一个 WM_SIZE 消息，都会重新构造小球。这就需要内存设备描述表与视频显示器兼容：

```
hdcMem = CreateCompatibleDC (hdc) ;
```

小球的直径设为窗口客户区高度或宽度中较短者的 1/16。不过，程序构造的位图却比小球大：从位图中心到位图 4 个边的距离是小球半径的 1.5 倍：

```
hBitmap = CreateCompatibleBitmap (hdc, cxTotal, cyTotal) ;
```

在将位图选进一个内存设备描述表后，整个位图的背景都会被设置成白色：

```
Rectangle (hdcMem, -1, -1, xTotal + 1, yTotal + 1) ;
```

这些奇数的坐标使矩形边框在位图之外着色。一个斜线画刷被选进内存设备描述表，并将小球画在位图的中央：

```
Ellipse (hdcMem, xMove, yMove, xTotal - xMove, yTotal - yMove) ;
```

当小球移动时，小球边界的空白会有效地删除前一时刻的小球图像。在另一个位置重画小球只需在 BitBlt 调用中使用 SRCCOPY 的 ROP 代码：

```
BitBlt (hdc, xCenter - cxTotal / 2, yCenter - cyTotal / 2, cxTotal, cyTotal,
        hdcMem, 0, 0, SRCCOPY) ;
```

BOUNCE 程序只是演示了在显示器上移动一个图像的最简单的方法，但这种方法对于一般情况并不能令人满意。如果您对动画感兴趣，那么您应该研究一些在源和目标中执行排他性的或操作的其他的 ROP 代码（例如 SRCINVERT）。其他动画技术涉及到 Windows 调色板（以及 AnimatePalette 函数）和 CreateDIBSection 函数。对于更高级的动画，您也许需要放弃 GDI，而使用 DirectX 接口了。

14.4.12 窗口外的位图

SCRAMBLE 程序，如程序 14-11 所示，编写非常粗糙，我本不应该演示，但它示范了一些有趣的技术，而且在交换两个显示矩形内容的 BitBlt 操作的过程中，用内存设备描述表作为临时存储空间。

程序 14-11　SCRAMBLE 程序

SCRAMBLE.C

```
/*-------------------------
   SCRAMBLE.C——Scramble (and Unscramble) Screen
               (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

#define NUM 300

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static int iKeep [NUM][4] ;
     HDC     hdcScr, hdcMem ;
     int     cx, cy ;
     HBITMAP hBitmap ;
     HWND    hwnd ;
     int     i, j, x1, y1, x2, y2 ;

     if (LockWindowUpdate (hwnd = GetDesktopWindow ()))
     {
          hdcScr  = GetDCEx (hwnd, NULL, DCX_CACHE | DCX_LOCKWINDOWUPDATE) ;
          hdcMem  = CreateCompatibleDC (hdcScr) ;
          cx      = GetSystemMetrics (SM_CXSCREEN) / 10 ;
          cy      = GetSystemMetrics (SM_CYSCREEN) / 10 ;
          hBitmap = CreateCompatibleBitmap (hdcScr, cx, cy) ;

          SelectObject (hdcMem, hBitmap) ;

          srand ((int) GetCurrentTime ()) ;

          for (i = 0 ; i < 2 ; i++)
          for (j = 0 ; j < NUM ; j++)
          {
               if (i == 0)
               {
                    iKeep [j] [0] = x1 = cx * (rand () % 10) ;
                    iKeep [j] [1] = y1 = cy * (rand () % 10) ;
                    iKeep [j] [2] = x2 = cx * (rand () % 10) ;
                    iKeep [j] [3] = y2 = cy * (rand () % 10) ;
               }
```

```
                else
                {
                    x1 = iKeep [NUM - 1 - j] [0] ;
                    y1 = iKeep [NUM - 1 - j] [1] ;
                    x2 = iKeep [NUM - 1 - j] [2] ;
                    y2 = iKeep [NUM - 1 - j] [3] ;
                }
                BitBlt (hdcMem, 0, 0, cx, cy, hdcScr, x1, y1, SRCCOPY) ;
                BitBlt (hdcScr, x1, y1, cx, cy, hdcScr, x2, y2, SRCCOPY) ;
                BitBlt (hdcScr, x2, y2, cx, cy, hdcMem, 0, 0, SRCCOPY) ;

                Sleep (10) ;
            }

            DeleteDC (hdcMem) ;
            ReleaseDC (hwnd, hdcScr) ;
            DeleteObject (hBitmap) ;

            LockWindowUpdate (NULL) ;
        }
        return FALSE ;
    }
```

SCRAMBLE 没有窗口过程。在 WinMain 中,它首先调用带有桌面窗口句柄的 LockWindowUpdate。此函数临时防止其他程序刷新屏幕,然后 SCRAMBLE 通过调用带有参数 DCX_LOCKWINDOWUPDATE 的 GetDCEx 来获得整个屏幕的设备描述表。这样就只有 SCRAMBLE 可以刷新屏幕了。

然后,SCRAMBLE 确定全屏幕的尺寸,并将长宽分别除以 10,程序用这个尺寸(名称是 cx 和 cy)来创建一个位图,并将该位图选进内存设备描述表。

使用 C 语言的 rand 函数,SCRAMBLE 计算出 4 个随机值(两个坐标点)作为 cx 和 cy 的倍数。程序通过三次调用 BitBlt 函数来交换两个矩形块中显示的内容。第一次将从第一个坐标点开始的矩形复制到内存设备描述表。第二次 BitBlt 将从第二坐标点开始的矩形复制到第一点开始的位置。第三次将内存设备描述表中的矩形复制到第二个坐标点开始的区域。

此过程将有效地交换显示器上两个矩形中的内容。SCRAMBLE 将执行 300 次交换,这时的屏幕显示肯定是一团糟。但不用担心,因为 SCRAMBLE 记得是怎么把显示弄得这样一团糟的,接着在退出前它会按相反的次序恢复原来的桌面显示(并解锁屏幕)!

您也可以用内存设备描述表将一个位图复制给另一个位图。例如,假定您要创建一个位图,该位图只包含另一个位图左上角的图形。如果原位图句柄为 hBitmap,那么您可以将其尺寸复制到一个 BITMAP 类型的结构中:

```
GetObject (hBitmap, sizeof (BITMAP), &bm) ;
```

然后创建一个未初始化的新位图,该位图的尺寸是原位图的 1/4:

```
hBitmap2 = CreateBitmap (bm.bmWidth / 2, bm.bmHeight / 2,
                         bm.bmPlanes, bm.bmBitsPixel, NULL) ;
```

现在创建两个内存设备描述表,并将原位图和新位图分别选进这两个内存设备描述表:

```
hdcMem1 = CreateCompatibleDC (hdc) ;
hdcMem2 = CreateCompatibleDC (hdc) ;

SelectObject (hdcMem1, hBitmap) ;
SelectObject (hdcMem2, hBitmap2) ;
```

最后,将第一个位图的左上角复制给第二个:

```
BitBlt (hdcMem2, 0, 0, bm.bmWidth / 2, bm.bmHeight / 2,
        hdcMem1, 0, 0, SRCCOPY) ;
```

剩下的只是清除工作:

```
DeleteDC (hdcMem1) ;
DeleteDC (hdcMem2) ;
DeleteObject (hBitmap) ;
```

BLOWUP.C 程序,如程序 14-12 所示,也用窗口更新锁定来在程序窗口之外显示一个捕获的矩形。此程序允许用户用鼠标捕获屏幕上的矩形区域,然后 BLOWUP 将该区域的内容复制到位图。在 WM_PAINT 消息期间,位图复制到程序的客户区,必要时将拉伸或压缩。(参见 675 页图 14-10。)

程序 14-12　BLOWUP 程序

BLOWUP.C

```
/*--------------------
     BLOWUP.C——Video Magnifier Program
                (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>
#include <stdlib.h>          // for abs definition
#include "resource.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName [] = TEXT ("Blowup") ;
```

```
     HACCEL      hAccel ;
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Blow-Up Mouse Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          if (! TranslateAccelerator (hwnd, hAccel, &msg))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     }
     return msg.wParam ;
}

void InvertBlock (HWND hwndScr, HWND hwnd, POINT ptBeg, POINT ptEnd)
{
```

```
    HDC hdc ;

    hdc = GetDCEx (hwndScr, NULL, DCX_CACHE | DCX_LOCKWINDOWUPDATE) ;
    ClientToScreen (hwnd, &ptBeg) ;
    ClientToScreen (hwnd, &ptEnd) ;
    PatBlt (hdc, ptBeg.x, ptBeg.y, ptEnd.x - ptBeg.x, ptEnd.y - ptBeg.y,
            DSTINVERT) ;
    ReleaseDC (hwndScr, hdc) ;
}

HBITMAP CopyBitmap (HBITMAP hBitmapSrc)
{
    BITMAP   bitmap ;
    HBITMAP  hBitmapDst ;
    HDC      hdcSrc, hdcDst ;

    GetObject (hBitmapSrc, sizeof (BITMAP), &bitmap) ;
    hBitmapDst = CreateBitmapIndirect (&bitmap) ;

    hdcSrc = CreateCompatibleDC (NULL) ;
    hdcDst = CreateCompatibleDC (NULL) ;

    SelectObject (hdcSrc, hBitmapSrc) ;
    SelectObject (hdcDst, hBitmapDst) ;

    BitBlt (hdcDst, 0, 0, bitmap.bmWidth, bitmap.bmHeight,
            hdcSrc, 0, 0, SRCCOPY) ;

    DeleteDC (hdcSrc) ;
    DeleteDC (hdcDst) ;

    return hBitmapDst ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BOOL      bCapturing, bBlocking ;
    static HBITMAP   hBitmap ;
    static HWND      hwndScr ;
    static POINT     ptBeg, ptEnd ;
    BITMAP           bm ;
    HBITMAP          hBitmapClip ;
    HDC              hdc, hdcMem ;
    int              iEnable ;
    PAINTSTRUCT      ps ;
    RECT             rect ;

    switch (message)
```

```
{
case WM_LBUTTONDOWN:
     if (!bCapturing)
     {
          if (LockWindowUpdate (hwndScr = GetDesktopWindow ()))
          {
               bCapturing = TRUE ;
               SetCapture (hwnd) ;
               SetCursor (LoadCursor (NULL, IDC_CROSS)) ;
          }
          else
               MessageBeep (0) ;
     }
     return 0 ;

case WM_RBUTTONDOWN:
     if (bCapturing)
     {
          bBlocking = TRUE ;
          ptBeg.x = LOWORD (lParam) ;
          ptBeg.y = HIWORD (lParam) ;
          ptEnd = ptBeg ;
          InvertBlock (hwndScr, hwnd, ptBeg, ptEnd) ;
     }
     return 0 ;

case WM_MOUSEMOVE:
     if (bBlocking)
     {
          InvertBlock (hwndScr, hwnd, ptBeg, ptEnd) ;
          ptEnd.x = LOWORD (lParam) ;
          ptEnd.y = HIWORD (lParam) ;
          InvertBlock (hwndScr, hwnd, ptBeg, ptEnd) ;
     }
     return 0 ;

case WM_LBUTTONUP:
     case WM_RBUTTONUP:
     if (bBlocking)
     {
          InvertBlock (hwndScr, hwnd, ptBeg, ptEnd) ;
          ptEnd.x = LOWORD (lParam) ;
          ptEnd.y = HIWORD (lParam) ;

          if (hBitmap)
          {
               DeleteObject (hBitmap) ;
               hBitmap = NULL ;
```

```
            }

            hdc = GetDC (hwnd) ;
            hdcMem = CreateCompatibleDC (hdc) ;
            hBitmap = CreateCompatibleBitmap (hdc,
                                abs (ptEnd.x - ptBeg.x),
                                abs (ptEnd.y - ptBeg.y)) ;

            SelectObject (hdcMem, hBitmap) ;

            StretchBlt (hdcMem, 0, 0, abs (ptEnd.x - ptBeg.x),
                                      abs (ptEnd.y - ptBeg.y),
                        hdc, ptBeg.x, ptBeg.y, ptEnd.x - ptBeg.x,
                                               ptEnd.y - ptBeg.y, SRCCOPY) ;

            DeleteDC (hdcMem) ;
            ReleaseDC (hwnd, hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
        }
        if (bBlocking || bCapturing)
        {
            bBlocking = bCapturing = FALSE ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
            ReleaseCapture () ;
            LockWindowUpdate (NULL) ;
        }
        return 0 ;

    case WM_INITMENUPOPUP:
        iEnable = IsClipboardFormatAvailable (CF_BITMAP) ?
                        MF_ENABLED : MF_GRAYED ;

        EnableMenuItem ((HMENU) wParam, IDM_EDIT_PASTE, iEnable) ;

        iEnable = hBitmap ? MF_ENABLED : MF_GRAYED ;

        EnableMenuItem ((HMENU) wParam, IDM_EDIT_CUT,    iEnable) ;
        EnableMenuItem ((HMENU) wParam, IDM_EDIT_COPY,   iEnable) ;
        EnableMenuItem ((HMENU) wParam, IDM_EDIT_DELETE, iEnable) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_EDIT_CUT:
        case IDM_EDIT_COPY:
            if (hBitmap)
```

```
            {
                hBitmapClip = CopyBitmap (hBitmap) ;
                OpenClipboard (hwnd) ;
                EmptyClipboard () ;
                SetClipboardData (CF_BITMAP, hBitmapClip) ;
            }
            if (LOWORD (wParam) == IDM_EDIT_COPY)
                return 0 ;
                                    // fall through for IDM_EDIT_CUT
        case IDM_EDIT_DELETE:
            if (hBitmap)
            {
                DeleteObject (hBitmap) ;
                hBitmap = NULL ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case IDM_EDIT_PASTE:
            if (hBitmap)
            {
                DeleteObject (hBitmap) ;
                hBitmap = NULL ;
            }
            OpenClipboard (hwnd) ;
            hBitmapClip = GetClipboardData (CF_BITMAP) ;

            if (hBitmapClip)
                hBitmap = CopyBitmap (hBitmapClip) ;

            CloseClipboard () ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hBitmap)
        {
            GetClientRect (hwnd, &rect) ;

            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;
            GetObject (hBitmap, sizeof (BITMAP), (PSTR) &bm) ;
            SetStretchBltMode (hdc, COLORONCOLOR) ;

            StretchBlt (hdc,    0, 0, rect.right, rect.bottom,
```

```
                    hdcMem, 0, 0, bm.bmWidth, bm.bmHeight, SRCCOPY) ;
            DeleteDC (hdcMem) ;
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (hBitmap)
            DeleteObject (hBitmap) ;

        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

BLOWUP.RC（摘录）

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
BLOWUP MENU DISCARDABLE
BEGIN
    POPUP "&Edit"
    BEGIN
        MENUITEM "Cu&t \ tCtrl + X",          IDM_EDIT_CUT
        MENUITEM "&Copy \ tCtrl + C",         IDM_EDIT_COPY
        MENUITEM "&Paste \ tCtrl + V ",       IDM_EDIT_PASTE
        MENUITEM "De&lete \ tDelete",         IDM_EDIT_DELETE
    END
END

//////////////////////////////////////////////////////////////////////////////
// Accelerator
BLOWUP ACCELERATORS DISCARDABLE
BEGIN
    "C",            IDM_EDIT_COPY,          VIRTKEY, CONTROL, NOINVERT
    "V ",           IDM_EDIT_PASTE,         VIRTKEY, CONTROL, NOINVERT
    VK_DELETE,      IDM_EDIT_DELETE,        VIRTKEY, NOINVERT
    "X",            IDM_EDIT_CUT,           VIRTKEY, CONTROL, NOINVERT
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
```

```
// Used by Blowup.rc

# define IDM_EDIT_CUT              40001
# define IDM_EDIT_COPY             40002
# define IDM_EDIT_PASTE            40003
# define IDM_EDIT_DELETE           40004
```

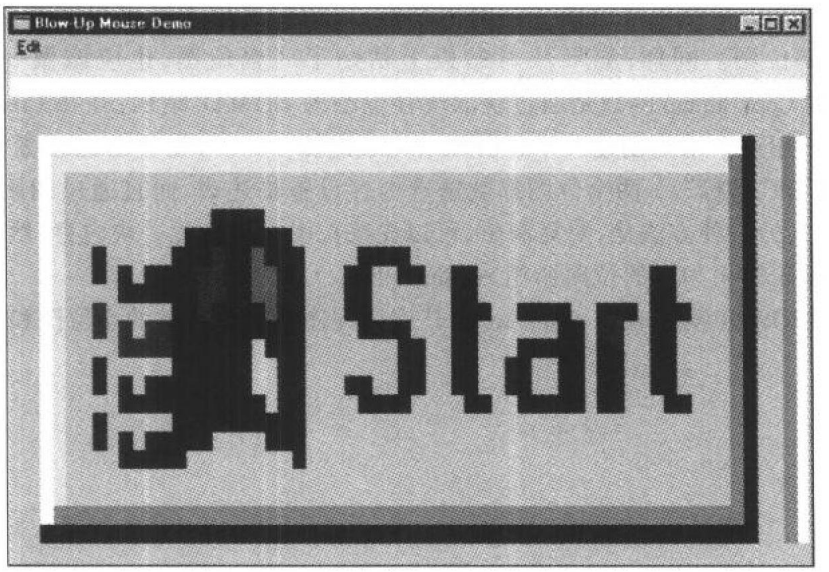


图 14-10　BLOWUP 显示的一个示例

由于鼠标捕获的限制，所以开始使用 BLOWUP 时会有些困难，需要逐渐适应。下面是使用本程序的方法：

1. 在 BLOWUP 客户区按住鼠标左键，鼠标指针会变成“+”字型。
2. 继续按住左键，将鼠标移到屏幕上的任何其他位置。鼠标光标的位置就是要捕获的矩形区域的左上角。
3. 继续按住左键，按下鼠标右键，然后拖动鼠标到要捕获的矩形区域的右下角。释放鼠标左键和右键。（释放鼠标左右键的次序无关紧要。）

鼠标光标恢复成箭头状，而捕获的矩形区域已复制到 BLOWUP 的客户区，并被适当的压缩或拉伸。

如果从右上角到左下角选取的话，那么 BLOWUP 将显示矩形区域的镜像。如果从左下到右上角选取，那么 BLOWUP 将显示颠倒的图像。如果从右下角至左上角选取，那么程序将综合这两种效果。

BLOWUP 还包含将位图复制到剪贴板，以及将剪贴板中的位图复制到程序的逻辑。

BLOWUP 处理 WM_INITMENUPOPUP 消息来启用或禁用 Edit 菜单中的不同项目，并通过 WM_COMMAND 消息来处理这些菜单项。您应该对这些代码的结构比较熟悉，因为它们与第十二章中的复制和粘贴文本项在本质上是一样的。

不过，对于位图，剪贴板项目不是全局句柄，而是位图句柄。当您使用 CF_BITMAP 时，GetClipboardData 函数返回一个 HBITMAP 对象，而且 SetClipboardData 函数接收一个 HBITMAP 对象。如果您想将位图传送给剪贴板，又想保留副本以供程序本身使用，那么您必须复制位图。同样，如果从剪贴板上粘贴了一幅位图，也应该做一个副本。BLOWUP 中的 CopyBitmap 函数是通过获取现存位图的 BITMAP 结构，并在 CreateBitmapIndirect 函数中用这个结构创建一个新位图来完成此操作的。（变量名的后缀 Src 和 Dst 分别代表“源”和“目标”。）两个位图都被选进内存设备描述表，而且通过调用 BitBlt 来传递位图位。（另一种方法是，要复制位，可以先按位图大小分配一块内存，然后为源位图调用 GetBitmapBits，为目的位图调用 SetBitmapBits。）

我发现 BLOWUP 对于检查 Windows 及其应用程序中大量分散的小位图和图片非常有用。

第十五章

与设备无关的位图

在上一章我们了解到 Windows GDI 位图对象(也称为与设备有关的位图,或 DDB)对于许多程序设计任务都很有用。但是我并没有演示把这些位图保存到磁盘文件中,或把它们重新加载到内存中的方法。这是以前在 Windows 中使用的方法,现在根本不做。因为位图的位格式高度依赖于设备,所以 DDB 不适用于图像交换。DDB 内没有色彩表来指定位图的位与色彩之间的联系。DDB 仅当在 Windows 会话的生存期内被创建和清除时才有意义。

Windows 3.0 中引入了与设备无关的位图(DIB),提供了适用于交换的图像文件格式。正如您所知的,像.GIF 或.JPEG 之类的其他图像文件格式在 Internet 上比 DIB 文件更常见。这主要是因为.GIF 和.JPEG 格式进行了压缩,显著地减少了下载的时间。尽管有一个用于 DIB 的压缩方案,但极少使用。DIB 内的位图几乎都没有被压缩。如果您想在程序中操作位图,这实际上是一个优点。DIB 不像.GIF 和.JPEG 文件,Windows API 直接支持 DIB。如果在内存中有 DIB,您就可以提供指向该 DIB 的指针作为某些函数的参数,来显示 DIB 或把 DIB 转化为 DDB。

15.1 DIB 文件格式

有意思的是,DIB 格式并不来源于 Windows。它首先定义在 OS/2 的 1.1 版中,该操作系统最初由 IBM 和 Microsoft 在 80 年代中期开始开发。OS/2 1.1 在 1988 年发布,并且是第一个包含了类似 Windows 的图形用户界面的 OS/2 版本,该图形用户界面被称为"显示管理器(PM)"。"显示管理器"包含了定义位图格式的"图形编程接口(GPI)"。

然后在 Windows 3.0 中(发布于 1990)使用了 OS/2 位图格式,这时称之为 DIB。Windows 3.0 也包含了原始 DIB 格式的变体,并在 Windows 下成为标准。在 Windows 95(及 Windows NT 4.0)和 Windows 98(及 Windows NT 5.0)下也定义了一些其他的增强功

能,我会在本章讨论它们。

DIB 首先作为一种文件格式,它的扩展名为.BMP,在极少数情况下为.DIB。Windows 应用程序使用的位图图像被当作 DIB 文件创建,并作为只读资源存储在程序的可执行文件中。图标和鼠标光标也是形式略微不同的 DIB 文件。

程序能将 DIB 文件减去前 14 个字节加载到连续的内存块中。这时就可以称它为"紧缩 DIB(packed-DIB)格式的位图"。在 Windows 下运行的应用程序可以使用紧缩 DIB 格式,通过 Windows 剪贴板来交换图像或创建画刷。程序也可以完全访问 DIB 的内容,并以任意方式修改 DIB。

程序也可以在内存中创建自己的 DIB,然后把它们存入文件。程序使用 GDI 函数调用就能"绘制"这些 DIB 内的图像,也能在进程中利用别的基于内存的 DIB 直接设置和操作像素位。

在内存中加载 DIB 后,程序也可以通过几个 Windows API 函数调用来使用 DIB 数据,我将在本章中讨论有关内容。与 DIB 相关的 API 调用是很少的,并且主要与视频显示器或打印机页面上显示 DIB 有关,还与转换 GDI 位图对象有关。

除了这些内容以外,还有许多应用程序需要完成的 DIB 任务,而这些任务 Windows 操作系统并不支持。例如,程序可能访问了 24 位 DIB,并想把它转化为带有最优化的 256 色调色板的 8 位 DIB,而 Windows 不会为您执行这些操作。但是在本章和下一章将向您显示 Windows API 之外的操作 DIB 的方式。

15.1.1　OS/2 风格的 DIB

先不要陷入太多的细节,让我们看一下与首先在 OS/2 1.1 中出现的位图格式兼容的 Windows DIB 格式。

DIB 文件有 4 个主要部分:

- 文件头;
- 信息头;
- RGB 色彩表(不一定有);
- 位图像素位。

您可以把前两部分看成是 C 的数据结构,把第三部分看成是数据结构的数组。在 Windows 头文件 WINGDI.H 中说明了这些结构。内存中的紧缩 DIB 格式有三个部分:

- 信息头;
- RGB 色彩表(不一定有);
- 位图像素位。

除了没有文件头外,其他部分与存储在文件内的 DIB 相同。

DIB文件(不是基于内存的紧缩DIB)以定义为如下结构的14个字节的文件头开始:

```
typedef struct tagBITMAPFILEHEADER     // bmfh
{
    WORD    bfType ;          // signature word "BM" or 0x4D42
    DWORD   bfSize ;          // entire size of file
    WORD    bfReserved1 ;     // must be zero
    WORD    bfReserved2 ;     // must be zero
    DWORD   bfOffsetBits ;    // offset in file of DIB pixel bits
}
BITMAPFILEHEADER, * PBITMAPFILEHEADER ;
```

在WINGDI.H内定义的结构可能与这不完全相同(注释是我加上的),但在功能上是相同的。第一个注释(就是文本"bmfh")指出了给这种数据类型的数据变量命名时推荐的缩写。如果在程序内看到了名为pbmfh的变量,这可能是一个指向BITMAPFILEHEADER类型结构的指针或指向PBITMAPFILEHEADER类型变量的指针。

结构的长度为14字节,它以两个字母"BM"开头以指明是位图文件。这是一个WORD值0x4D42。紧跟在"BM"后的DWORD以字节为单位,指出了包括文件头在内的文件大小。下两个WORD字段设置为0。(在与DIB文件格式相似的鼠标光标文件内,这两个字段指出光标的"热点(hot spot)"。)结构还包含一个DWORD字段,它指出了文件中像素位开始位置的字节偏移量。此数来源于DIB信息头中的信息,为了使用的方便而在此处提供。

在OS/2风格的DIB内,BITMAPFILEHEADER结构后紧跟BITMAPCOREHEADER结构,它提供了关于DIB图像的基本信息。紧缩的DIB开始于BITMAPCOREHEADER:

```
typedef struct tagBITMAPCOREHEADER     // bmch
{
    DWORD   bcSize ;          // size of the structure = 12
    WORD    bcWidth ;         // width of image in pixels
    WORD    bcHeight ;        // height of image in pixels
    WORD    bcPlanes ;        // = 1
    WORD    bcBitCount ;      // bits per pixel (1, 4, 8, or 24)
}
BITMAPCOREHEADER, * PBITMAPCOREHEADER ;
```

单词"core(核心)"在此处看起来有点奇特,它意味着这种格式是由此派生出的其他位图格式的基础(故称为核心)。

BITMAPCOREHEADER结构中的bcSize字段指出了数据结构的大小,此情况下为12字节。

bcWidth和bcHeight字段包含了以像素为单位的位图大小。尽管为这些字段使用WORD意味着一个DIB可能为65 535像素高和宽,但是几乎不会用到那么大的单位。

bcPlanes 字段的值始终是 1。这个字段是我们在上一章中遇到的早期 Windows GDI 位图对象的残留物。

bcBitCount 字段指出了每像素的位数。对于 OS/2 风格的 DIB,这可能是 1、4、8,或 24。DIB 图像中的颜色数等于 $2^{bmch.bcBitCount}$,或用 C 的语法表示为

```
1 << bmch.bcBitCount
```

因此,bcBitCount 字段等于:

- 1(对于 2 色 DIB);
- 4(对于 16 色 DIB);
- 8(对于 256 色 DIB);
- 24(对于全色 DIB)。

当我提到"8 位 DIB"时,就是说每像素占 8 位的 DIB。

对于前三种情况(也就是位数为 1、4 和 8 时),BITMAPCOREHEADER 后紧跟色彩表,24 位 DIB 没有色彩表。色彩表是一个 3 字节 RGBTRIPLE 结构的数组,数组中的每个元素代表图像中的每种颜色:

```
typedef struct tagRGBTRIPLE     // rgbt
{
     BYTE rgbtBlue ;      // blue level
     BYTE rgbtGreen ;     // green level
     BYTE rgbtRed ;       // red level
}
RGBTRIPLE ;
```

这样排列色彩表以便 DIB 中最重要的颜色首先显示,我们将在下一章说明原因。

WINGDI.H 头文件也定义了下面的结构:

```
typedef struct tagBITMAPCOREINFO     // bmci
{
     BITMAPCOREHEADER bmciHeader ;          // core-header structure
     RGBTRIPLE        bmciColors[1] ;       // color table array
}
BITMAPCOREINFO, * PBITMAPCOREINFO ;
```

这个结构把信息头与色彩表结合起来。虽然在这个结构中 RGBTRIPLE 结构的数量等于 1,但在 DIB 文件内您绝对不会发现只有一个 RGBTRIPLE。根据每个像素的位数,色彩表的大小始终是 2、16 或 256 个 RGBTRIPLE 结构。如果需要为 8 位 DIB 分配 PBITMAPCOREINFO 结构,您可以这样做:

```
pbmci = malloc (sizeof (BITMAPCOREINFO) + 255 * sizeof (RGBTRIPLE)) ;
```

然后可以这样访问 RGBTRIPLE 结构:

```
pbmci -> bmciColors[i]
```

因为 RGBTRIPLE 结构的长度是 3 字节，许多 RGBTRIPLE 结构可能在 DIB 中以奇数地址开始。然而，因为在 DIB 文件内始终有偶数的 RGBTRIPLE 结构，所以紧跟在色彩表数组后的数据块总是以 WORD 地址边界开始。

紧跟在色彩表(24 位 DIB 中是信息头)后的数据是像素位本身。

15.1.2 从下向上

像大多数位图格式一样，DIB 中的像素位是按水平行组织的，用视频显示器硬件的术语称作“扫描线”。行数等于 BITMAPCOREHEADER 结构的 bcHeight 字段。然而，与大多数位图格式不同的是，DIB 从图像的底行开始，往上扫过图像。

在此应定义一些术语，当我们说“顶行”和“底行”时，指的是当其正确显示在显示器或打印机的页面上时出现在虚拟图像的顶部和底部。肖像的顶行是头发，底行是下巴。“第一行”指的是 DIB 文件的色彩表后的像素行，“最后行”指的是文件最末端的像素行。

因此，在 DIB 中，图像的底行是文件的第一行，图像的顶行是文件的最后一行。这被称为从下向上组织。因为这种组织和直觉相反，您可能会问：为什么要这样做?

好，现在我们回到 OS/2 的显示管理器。IBM 的某人认为 PM 内的坐标系统——包括窗口、图形和位图——应该是一致的。这引起了争论：大多数人，包括在全屏文本方式下编程和窗口环境下工作的程序员，认为应使用垂直坐标沿屏幕向下增加的坐标；然而，计算机图形程序员认为应使用解析几何的数学方法进行视频显示，这是一个垂直坐标在空间中向上增加的直角(或笛卡儿)坐标系。

简而言之，数学方法赢了。PM 内的所有事物都以左下角为原点(包括窗口坐标)，因此 DIB 也就有了那种方式。

15.1.3 DIB 像素位

DIB 文件的最后部分(在大多数情况下是 DIB 文件的主体)由实际的 DIB 的像素位组成。像素位是由从图像的底行开始，并沿着图像向上增长的水平行组织的。

DIB 中的行数等于 BITMAPCOREHEADER 结构的 bcHeight 字段。每一行的像素数等于该结构的 bcWidth 字段。每一行从最左边的像素开始，直到图像的右边。每个像素的位数可以从 bcBitCount 字段获取，为 1、4、8 或 24。

以字节为单位的每行长度始终是 4 的倍数。行的长度可以这样计算：

```
RowLength = 4 * ((bmch.bcWidth * bmch.bcBitCount + 31) / 32) ;
```

或者在 C 语言中用更有效的方法：

```
RowLength = ((bmch.bcWidth * bmch.bcBitCount + 31) & 31) >> 3 ;
```

如果需要,可通过在右边补充行(通常是用0)来完成长度。像素数据的总字节数等于RowLength 和 bmch.bcHeight 的乘积。

要了解像素编码的方式,让我们分别考虑4种情况。在下面的图表中,每个字节的位显示在框内并且编了号,7 表示最高位,0 表示最低位。像素也从行的最左端从0开始编号。

对于每像素1位的DIB,每字节对应为8像素。最左边的像素是第一个字节的最高位:

```
像素: 0 1 2 3 4 5 6 7  8 9 10 11 12 13 14 15  16 17 18 19 20 21 22 23
      7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0
```

每个像素可以是0或1。0位表示该像素的颜色由色彩表中第一个 RGBTRIPLE 条目给出。1位表示像素的颜色由色彩表的第二个条目给出。

对于每像素4位的DIB,每个字节对应两个像素。最左边的像素是第一个字节的高4位,以此类推:

```
像素:  —0—   —1—   —2—   —3—   —4—   —5—
      7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0
```

每像素4位的值的范围从0到15。此值是指向色彩表中16个条目的索引。

对于每像素8位的DIB,每个字节为1个像素:

```
像素:      —0—          —1—          —2—
      7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0
```

字节的值从0到255。同样,这也是指向色彩表中256个条目的索引。

对于每像素24位的DIB,每个像素需要3个字节来代表红、绿和蓝的颜色值。像素位的每一行,基本上就是 RGBTRIPLE 结构的数组,可能需要在每行的末尾补0,以使该行为4字节的倍数:

```
像素:     — 蓝 —          — 绿 —          — 红 —
      7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0
```

每像素24位的DIB没有色彩表。

15.1.4　扩展的 Windows DIB

现在我们掌握了 Windows 3.0 中介绍的与 OS/2 兼容的 DIB,同时我们也看一看 Windows 中 DIB 的扩展版本。

这种 DIB 形式跟前面的格式一样,以 BITMAPFILEHEADER 结构开始,但是接着是

BITMAPINFOHEADER 结构，而不是 BITMAPCOREHEADER 结构：

```
typedef struct tagBITMAPINFOHEADER     // bmih
{
     DWORD   biSize ;              // size of the structure = 40
     LONG    biWidth ;             // width of the image in pixels
     LONG    biHeight ;            // height of the image in pixels
     WORD    biPlanes ;            // = 1
     WORD    biBitCount ;          // bits per pixel (1, 4, 8, 16, 24, or 32)
     DWORD   biCompression ;       // compression code
     DWORD   biSizeImage ;         // number of bytes in image
     LONG    biXPelsPerMeter ;     // horizontal resolution
     LONG    biYPelsPerMeter ;     // vertical resolution
     DWORD   biClrUsed ;           // number of colors used
     DWORD   biClrImportant ;      // number of important colors
}
BITMAPINFOHEADER, * PBITMAPINFOHEADER ;
```

您可以通过检查结构的第一字段区分与 OS/2 兼容的 DIB 和 Windows DIB，前者为 12，后者为 40。

您将注意到，在这个结构内有 6 个附加的字段，但是 BITMAPINFOHEADER 不是简单地由 BITMAPCOREHEADER 加上一些新字段而成。仔细看一下：在 BITMAPCOREHEADER 结构中，bcWidth 和 bcHeight 字段是 16 位 WORD 值；而在 BITMAPINFOHEADER 结构中它们是 32 位 LONG 值。这是一个令人讨厌的小变化，当心它会给您带来麻烦。

另一个变化是：对于使用 BITMAPINFOHEADER 结构的 1 位、4 位和 8 位 DIB，色彩表不是 RGBTRIPLE 结构的数组。相反，BITMAPINFOHEADER 结构紧跟着一个 RGBQUAD 结构的数组：

```
typedef struct tagRGBQUAD     // rgb
{
     BYTE rgbBlue ;        // blue level
     BYTE rgbGreen ;       // green level
     BYTE rgbRed ;         // red level
     BYTE rgbReserved ;    // = 0
}
RGBQUAD ;
```

除了包括总是设置为 0 的第 4 个字段外，与 RGBTRIPLE 结构相同。WINGDI.H 头文件也定义了以下结构：

```
typedef struct tagBITMAPINFO  // bmi
{
     BITMAPINFOHEADER bmiHeader ;        // info-header structure
     RGBQUAD          bmiColors[1] ;     // color table array
}
BITMAPINFO, * PBITMAPINFO ;
```

注意，如果 BITMAPINFO 结构以 32 位的地址边界开始，因为 BITMAPINFOHEADER 结构的长度是 40 字节，所以 RGBQUAD 数组内的每一个条目也以 32 位边界开始。这样就确保通过 32 位微处理器能更有效地对色彩表数据寻址。

尽管 BITMAPINFOHEADER 最初是在 Windows 3.0 中定义的，但是许多字段在 Windows 95 和 Windows NT 4.0 中又重新定义了，并且被带入 Windows 98 和 Windows NT 5.0 中。比如现在的文档中说："如果 biHeight 是负数，则位图是自顶向下的 DIB，原点在左上角"。这很好，但是在 1990 年刚开始定义 DIB 格式时，如果有人做了这个决定，那会更好。我的建议是避免创建自顶向下的 DIB。有一些程序在编写时没有考虑这种新"特性"，在遇到负的 biHeight 字段时会崩溃。还有如 Microsoft Word 97 带有的 Microsoft Photo Editor 在遇到从顶向下的 DIB 时会报告"图像高度非法"（虽然 Word 97 本身不会出错）。

biPlanes 字段始终是 1，但 biBitCount 字段现在可以是 16 或 32，以及 1、4、8 或 24。这也是 Windows 95 和 Windows NT 4.0 的新特性。一会儿我将介绍这些附加格式工作的方式。

现在让我们先跳过 biCompression 和 biSizeImage 字段，一会儿再讨论它们。

biXPelsPerMeter 和 biYPelsPerMeter 字段以每米多少像素这种笨拙的单位指出图像的实际尺寸。（"pel"——picture element（图像元素）——是 IBM 对像素的称呼。）Windows 在内部不使用此类信息。然而，应用程序能够利用它以准确的大小显示 DIB。如果 DIB 来源于没有方形像素的设备，这些字段是很有用的。在大多数 DIB 内，这些字段设置为 0，这表示没有建议的实际大小。每英寸 72 点的分辨率（有时用于视频显示器，尽管实际分辨率依赖于显示器的大小）大约相当于每米 2835 个像素，300 DPI 的普通打印机的分辨率是每米 11 811 个像素。

biClrUsed 是非常重要的字段，因为它影响色彩表中条目的数量。对于 4 位和 8 位 DIB，它能分别指出色彩表中包含了小于 16 或 256 个条目。虽然并不常用，但这是一种缩小 DIB 大小的方法。例如，假设 DIB 图像仅包括 64 个灰度，biClrUsed 字段设置为 64，并且色彩表包含了 64 个 RGBQUAD 结构，以对应 256 字节的色彩表总规模。像素值的范围从 0x00 到 0x3F。DIB 仍然每像素需要 1 字节，但每个像素字节的高 2 位为 0。如果 biClrUsed 字段设置为 0，意味着色彩表包含了由 biBitCount 字段表示的全部条目数。

从 Windows 95 开始，biClrUsed 字段对于 16 位、24 位或 32 位 DIB 可以为非 0。在这些情况下，Windows 不使用色彩表解释像素位。相反，它指出 DIB 中色彩表的大小，程序使用该信息来设置调色板在 256 色视频显示器上显示 DIB。您可能想起在 OS/2 兼容格式中，24 位 DIB 没有色彩表。在 Windows 3.0 中的扩展格式中，也与这一样。而在 Windows 95 中，24 位 DIB 有色彩表，biClrUsed 字段指出了它的大小。

总结如下：

◆ 对于 1 位 DIB，biClrUsed 始终是 0 或 2。色彩表始终有两个条目。

- 对于 4 位 DIB，如果 biClrUsed 字段是 0 或 16，则色彩表有 16 个条目。如果是从 2 到 15 的数，则指的是色彩表中的条目数。每个像素的最大值比该数小 1。
- 对于 8 位 DIB，如果 biClrUsed 字段是 0 或 256，则色彩表有 256 个条目。如果是从 2 到 225 的数，则指的是色彩表中的条目数。每个像素的最大值比该数小 1。
- 对于 16 位、24 位或 32 位 DIB，biClrUsed 字段通常为 0。如果它不为 0，则指的是色彩表中的条目数。运行于 256 色视频适配器的应用程序能使用这些条目来为 DIB 设置调色板。

另一个警告：原先使用早期 DIB 文档编写的程序在 24 位 DIB 中没有色彩表，如果在程序里应用它的话要冒一定的风险。

biClrImportant 字段实际上没有 biClrUsed 字段重要，它通常被设置为 0，以指出色彩表中所有的颜色都是重要的，或者它与 biClrUsed 有相同的值。两种方法意味着一件事，如果它被设置为 0 与 biClrUsed 之间的值，就意味着 DIB 图像能仅仅通过色彩表中第一个 biClrImportant 条目合理地获取。当在 256 色视频适配器上并排显示两个或更多 8 位 DIB 时，这是很有用的。

对于 1 位、4 位、8 位和 24 位的 DIB，像素位的组织和与 OS/2 兼容的 DIB 是相同的，一会儿我将讨论 16 位和 32 位 DIB。

15.1.5　现实性检查

当遇到一个由其他程序或别人创建的 DIB 时，您希望从中发现什么内容呢？

尽管在首次发布 Windows 3.0 时，OS/2 风格的 DIB 已经很普遍，但最近它们已经很少出现了。许多程序员在编写快速 DIB 例程时实际上忽略了它们。您遇到的任何 4 位 DIB 可能是 Windows 的“画图”程序使用 16 色视频显示器创建的，在这些显示器上色彩表具有标准的 16 种颜色。

最普遍的 DIB 可能是每像素 8 位，8 位 DIB 分为两类：灰度 DIB 和混色 DIB。遗憾的是，头信息中并没有指出 8 位 DIB 的类型。

许多灰度 DIB 有一个等于 64 的 biClrUsed 字段，指出色彩表中的 64 个条目。这些条目通常以上升的灰度级排列，也就是说色彩表以 00-00-00、04-04-04、08-08-08、0C-0C-0C 的 RGB 值开始，并包括 F0-F0-F0、F4-F4-F4、F8-F8-F8 和 FC-FC-FC 的 RGB 值。此类色彩表可用如下的公式计算：

```
rgb[i].rgbRed = rgb[i].rgbGreen = rgb[i].rgbBlue = i * 256 / 64 ;
```

在这里，rgb 是 RGBQUAD 结构的数组，i 的范围从 0 到 63。灰度色彩表可用如下的公式计算：

```
rgb[i].rgbRed = rgb[i].rgbGreen = rgb[i].rgbBlue = i * 255 / 63 ;
```

因此该表以 FF-FF-FF 结尾。

实际上使用哪个公式都没有什么关系。许多视频显示适配器和显示器没有比 6 位更大的色彩精度。第一个公式承认了这个事实,然而当产生小于 64 的灰度时——可能是 16 或 32(在此情况下公式的除数分别是 15 和 31)——使用第二个公式更适合,因为它确保了色彩表的最后一个条目是 FF-FF-FF,即白色。

当某些 8 位灰度 DIB 在色彩表内有 64 个条目时,其他灰度的 DIB 会有 256 个条目。biClrUsed 字段实际上可以为 0(指出色彩表中有 256 个条目),或者是从 2 到 256 的数。当然,若 biClrUsed 值是 2 的话,就没有任何意义(因为这样的 8 位 DIB 能当作 1 位 DIB 被重新编码),或者小于或等于 16 的值也没意义(因为它能当作 4 位 DIB 被重新编码)。任何情况下,色彩表中的条目数必须与 biClrUsed 字段相同(如果 biClrUsed 是 0 则是 256),并且像素值不能超过色彩表条目数减 1 的值。这是因为像素值是指向色彩表数组的索引。对于 biClrUsed 值为 64 的 8 位 DIB,像素值的范围从 0x00 到 0x3F。

在这里应记住一件重要的事情:当 8 位 DIB 具有由整个灰度组成的色彩表(也就是说,当红色、绿色和蓝色程度相等时),或当这些灰度级在色彩表中递增(像上面描述的那样)时,像素值自身就代表了灰色的程度。也就是说,如果 biClrUsed 是 64,那么 0x00 像素值为黑色,0x20 的像素值是 50%的灰度,0x3F 的像素值为白色。

这对于一些图像处理任务是很重要的,因为您可以完全忽略色彩表,仅需处理像素值。这是很有用的,如果允许回到过去,并对 BITMAPINFOHEADER 结构做一个简单的更改,我会添加一个标志,指出 DIB 映像是灰度的,那么 DIB 就没有色彩表,并且像素值直接指出了灰度。

混色的 8 位色的 DIB 一般使用整个色彩表,它的 biClrUsed 字段为 0 或 256。然而您也能遇到较小的颜色数,如 236。我们应承认一个事实:程序通常只能在 Windows 颜色板内更改 236 个条目以正确显示这些 DIB,我将在下章讨论这些内容。

biXPelsPerMeter 和 biYPelsPerMeter 很少为非 0 值,biClrImportant 字段不为 0 或 biClrUsed 值的情况也很少。

15.1.6　DIB 压缩

前面没有讨论 BITMAPINFOHEADER 中的 biCompression 和 biSizeImage 字段,现在我们讨论一下这些值。

biCompression 字段可以为 4 个常量之一,它们是:BI_RGB、BI_RLE8、BI_RLE4 或 BI_BITFIELDS,它们定义在 WINGDI.H 头文件中,值分别为 0,1,2,3。此字段有两个用途:对于 4 位和 8 位 DIB,它指出像素位用一种行程(run-length)编码方式压缩了;对于 16 位和 32 位 DIB,它指出了颜色掩蔽(color masking)是否用于对像素位进行编码。这两个特性都是在 Windows 95 中引入的。

首先让我们看一下 RLE 压缩：

- 对于 1 位 DIB，biCompression 字段始终是 BI_RGB；
- 对于 4 位 DIB，biCompression 字段可以是 BI_RGB 或 BI_RLE4；
- 对于 8 位 DIB，biCompression 字段可以是 BI_RGB 或 BI_RLE8；
- 对于 24 位 DIB，biCompression 字段始终是 BI_RGB。

如果值是 BI_RGB，像素位存储的方式和 OS/2 兼容的 DIB 一样，否则就使用行程编码压缩像素位。

行程编码(RLE)是一种最简单的数据压缩形式，它是根据 DIB 映像在一行内经常有相同的像素串这个事实进行压缩的。RLE 通过对重复像素的值及重复的次数编码来节省空间，而用于 DIB 的 RLE 方案只定义了很少的矩形 DIB 图像，也就是说，矩形的某些区域是未定义的，这能被用于表示非矩形图像。

8 位 DIB 的行程编码在概念上更简单一些，因此让我们从这里入手。表 15-1 会帮助您理解当 biCompression 字段等于 BI_RGB8 时像素位的编码方式。

表 15-1

字节 1	字节 2	字节 3	字节 4	含义
00	00			行结束
00	01			图像结束
00	02	dx	dy	移到(x + dx,y + dy)
00	n = 03 ~ FF			使用下面 n 个像素
n = 01 ~ FF	像素			重复像素 n 次

当对压缩的 DIB 解码时，可成对查看 DIB 数据字节，例如表 15-1 内的“字节 1”和“字节 2”。表格以这些字节值的递增方式排列，但从下向上讨论这个表格会更有意义。

如果第一个字节非 0(表 15-1 最后一行的情况)，那么它就是行程的重复因子。随后的像素值被重复多次，例如，字节对

0x05 0x27

解码后的像素值为

0x27 0x27 0x27 0x27 0x27

当然，DIB 会有许多数据不是像素到像素的重复，表 15-1 倒数第二行处理这种情况，它指出紧跟着的像素数应逐个使用。例如，考虑序列

0x00 0x06 0x45 0x32 0x77 0x34 0x59 0x90

解码后的像素值为

0x45 0x32 0x77 0x34 0x59 0x90

这些序列总是以 2 字节边界排列。如果第二个字节是奇数,那么序列内就有一个未使用的多余字节。例如,序列

0x00 0x05 0x45 0x32 0x77 0x34 0x59 0x00

解码后的像素值为

0x45 0x32 0x77 0x34 0x59

这就是行程编码的工作方式。很明显,如果在 DIB 图像内没有重复的像素,使用此压缩技术实际上增加了 DIB 文件的大小。

上面表 15-1 的前三行指出了矩形 DIB 图像的某些部分可以不用定义的方法。想象一下,您写的程序对已压缩的 DIB 进行解压缩,在这个解压缩的例程中,您将保持一对数字(x,y),开始为(0,0)。每对一个像素解码,x 的值就增加 1,每完成一行就将 x 重新置为 0,并且增加 y 的值。

当遇到字节 0x00 后面跟着 0x02 时,读取下两个字节并把它们作为无符号的增量添加到当前的 x 和 y 值中,然后继续解码。当遇到 0x00 字节后面跟着 0x00 时,就解完了一行,应将 x 置 0 并增加 y 值。当遇到 0x00 字节后面跟着 0x01 时,就完成解码了。这些代码允许 DIB 包含那些未定义的区域,它们用于对非矩形图像编码或在制作数字动画和电影时非常有用(因为几乎每一帧都有来自前一帧的信息,并且不需重新编码)。

对于 4 位 DIB,编码一般是相同的,但更复杂,因为字节和像素之间不是一对一的关系。

如果读取的第一个字节非 0,那就是一个重复因子 n。第二个字节(被重复的)包含 2 个像素,在 n 个像素的被解码的序列中交替出现,例如,字节对

0x07 0x35

被解码为

0x35 0x35 0x35 0x3?

其中的问号指出像素还未知,如果是上面显示的 0x07 0x35 对紧跟着的字节对

0x05 0x24

则整个解码的序列为

0x35 0x35 0x35 0x32 0x42 0x42

如果字节对中的第一字节是 0x00 ,第二个字节是 0x03 或更大,则使用第二字节指出的像素数。例如,序列

0x00 0x05 0x23 0x57 0x10 0x00

解码为

0X23 0X57 0X1?

注意:必须填补解码的序列使其成为偶数字节。

无论 biCompression 字段是 BI_RLE4 或 BI_RLE8,biSizeImage 字段都指出了字节内 DIB 像素数据的大小。如果 biCompression 字段是 BI_RGB,则 biSizeImage 通常为 0,但是它能被设置为行内字节长度的 biHeight 倍,就像在本章前面计算的那样。

当前文档说"自顶向下的 DIB 不能被压缩。"自顶向下的 DIB 是在 biHeight 字段为负数的情况下出现的。

15.1.7 颜色掩蔽

biCompression 字段也用于连接 Windows 95 中新出现的 16 位和 32 位 DIB。对于这些 DIB,biCompression 字段可以是 BI_RGB 或 BI_BITFIELDS(均定义为值 3)。

让我们看一下 24 位 DIB 的像素格式,它始终有一个等于 BI_RGB 的 biCompression 字段:

```
像素:     — 蓝 —          — 绿 —          — 红 —
   7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0
```

也就是说,每一行基本上都是 RGBTRIPLE 结构的数组,在每行末端有可能补充值,以使行内的字节是 4 的倍数。

对于具有 biCompression 字段等于 BI_RGB 的 16 位 DIB,每个像素需要两个字节。颜色是这样来编码的:

```
像素: …绿 — —— 蓝 ——— 0 — 红 —— 绿…
     7 6 5 4 3 2 1 0  7 6 5 4 3 2 1 0
```

每种颜色使用 5 位。对于行内的第一个像素,蓝色值是第一个字节的最低 5 位。绿色值在第一和第二个字节中都有位:绿色值的两个最高位是第二个字节中的两个最低位,绿色值的 3 个最低位是第一个字节中的 3 个最高位。红色值是第二个字节中的 2 到 6 位。第二个字节的最高位是 0。

当以 16 位字访问像素值时,这会更加有意义。因为多个字节值的最低位首先被存储,像素字如下:

```
      0  — 红 —   — 绿 —  — 蓝 —
像素字: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
```

假设在 wPixel 内存储了 16 位像素，可以用下列公式计算红色、绿色和蓝色值：

```
Red   = ((0x7C00 & wPixel) >> 10) << 3 ;
Green = ((0x03E0 & wPixel) >> 5) << 3 ;
Blue  = ((0x001F & wPixel) >> 0) << 3 ;
```

首先，使用掩码值与像素进行了按位与操作。此结果是：红色向右移动 10 位，绿色向右移动 5 位，蓝色向右移动 0 位。（这些移动值我称为“右移值”。）这就产生了从 0x00 到 0x1F 的颜色值，这些值必须向左移动 3 位，以合成从 0x00 到 0xF8 的颜色值。（这些移动值我称为“左移值”。）

请记住：如果 16 位 DIB 的像素宽度是奇数，每行会在末端补充多余的 2 字节，以使字节长度能被 4 整除。

对于 32 位 DIB，如果 biCompression 等于 BI_RGB，每个像素需要 4 字节。蓝色值是第一个字节，绿色为第二个，红色为第三个，第四字节等于 0。也可这么说，像素是 RGBQUAD 结构的数组。因为每个像素的长度是 4 个字节，所以在行末端就不需填补字节。

若想以 32 位双字访问每个像素，它就像这样：

```
          ——— 0 ———  ——  红  ——  ——  绿  ——  ——  蓝  ——
像素双字：31 30 ... 25 24 23 22 ... 17 16 15 14 ... 9  8  7  6  ... 1  0
```

或者，如果 dwPixel 是 32 位双字，

```
Red   = ((0x00FF0000 & dwPixel) >> 16) << 0 ;
Green = ((0x0000FF00 & dwPixel) >> 8) << 0 ;
Blue  = ((0x000000FF & dwPixel) >> 0) << 0 ;
```

左移值全为 0，因为颜色值在 0xFF 已是最大。注意这个双字与 Windows GDI 函数调用中用于指定 RGB 颜色的 32 位 COLORREF 值不一致。在 COLORREF 值中，红色占最低位的字节。

到目前为止，我们讨论了当 biCompression 字段为 BI_RGB 时，16 位和 32 位 DIB 的默认情况。如果 biCompression 字段为 BI_BITFIELDS，则紧跟着 DIB 的 BITMAPINFOHEADER 结构的是三个 32 位颜色掩码，第一个用于红色，第二个用于绿色，第三个用于蓝色。可以使用 C 的按位与操作符(&)把这些掩码应用于 16 位或 32 位的像素值上，然后通过右移值向右移动结果，这些值只有检查完掩码后才能知道。颜色掩码的规则应该很明确：每个颜色掩码内的 1 位必须是连续的，并且 1 位不能在三个掩码中重叠。

让我们举个例子，如果有一个 16 位 DIB，并且 biCompression 字段为 BI_BITFIELDS。您应该检查 BITMAPINFOHEADER 结构之后的前三个双字：

```
0x0000F800
```

```
0x000007E0
0x0000001F
```

注意：因为这是 16 位 DIB，所以只有位于底部 16 位的位才能被设置为 1。您可以把变量 dwMask[0]、dwMask[1]和 dwMask[2]设置为这些值。现在可以编写从掩码中计算右移和左移值的一小段例程了：

```
int MaskToRShift (DWORD dwMask)
{
     int iShift ;

     if (dwMask == 0)
          return 0 ;

     for (iShift = 0 ; !(dwMask & 1) ; iShift ++ )
          dwMask >> = 1 ;

     return iShift ;
}

int MaskToLShift (DWORD dwMask)
{
     int iShift ;

     if (dwMask == 0)
          return 0 ;

     while (!(dwMask & 1))
          dwMask >> = 1 ;

     for (iShift = 0 ; dwMask & 1 ; iShift ++ )
          dwMask >> = 1 ;

     return 8 - iShift ;
}
```

然后调用 MaskToRShift 函数三次以获得右移值：

```
iRShift[0] = MaskToRShift (dwMask[0]) ;
iRShift[1] = MaskToRShift (dwMask[1]) ;
iRShift[2] = MaskToRShift (dwMask[2]) ;
```

分别得到值 11、5 和 0。然后调用 MaskToLShift：

```
iLShift[0] = MaskToLShift (dwMask[0]) ;
iLShift[1] = MaskToLShift (dwMask[1]) ;
iLShift[2] = MaskToLShift (dwMask[2]) ;
```

分别得到值 3、2 和 3。现在可以从像素中提取每种颜色：

```
Red   = ((dwMask[0] & wPixel) >> iRShift[0]) << iLShift[0] ;
Green = ((dwMask[1] & wPixel) >> iRShift[1]) << iLShift[1] ;
Blue  = ((dwMask[2] & wPixel) >> iRShift[2]) << iLShift[2] ;
```

除了颜色标记能大于 0x0000FFFF(这是 16 位 DIB 的最大掩码值)之外,过程与 32 位 DIB 一样。

注意:对于 16 位或 32 位 DIB,红色、绿色和蓝色值可以大于 255。实际上,在 32 位 DIB 中,如果掩码中有两个为 0,第三个应为 32 位颜色值 0xFFFFFFFF。当然,这有点荒唐,但不用担心这个问题。

不像 Windows NT, Windows 95 和 Windows 98 在使用颜色掩码时有许多限制。可用的值显示在表 15-2 中。

表 15-2

	16 位 DIB	**16 位 DIB**	**32 位 DIB**
红色掩码	0x00007C00	0x0000F800	0x00FF0000
绿色掩码	0x000003E0	0x000007E0	0x0000FF00
蓝色掩码	0x0000001F	0x0000001F	0x000000FF
速记为	5-5-5	5-6-5	8-8-8

换句话说,可以使用当 biCompression 为 BI_RGB 时所默认的两组掩码,加上我在前面例子中给出的掩码组。表格底行显示了一个速记符号来指出每像素红色、绿色和蓝色的位数。

15.1.8　第 4 版的 Header

我说过,Windows 95 更改了一些原始 BITMAPINFOHEADER 字段的定义。Windows 95 也包括了一个称为“BITMAPV4HEADER 的新的扩展的信息头”。如果您知道 Windows 95 曾被称为“Windows 4.0”,就会明白此结构的名称了。Windows NT 4.0 也支持此结构。

```
typedef struct
{
    DWORD       bV4Size ;           // size of the structure = 120
    LONG        bV4Width ;          // width of the image in pixels
    LONG        bV4Height ;         // height of the image in pixels
    WORD        bV4Planes ;         // = 1
    WORD        bV4BitCount ;       // bits per pixel (1, 4, 8, 16, 24, or 32)
    DWORD       bV4Compression ;    // compression code
    DWORD       bV4SizeImage ;      // number of bytes in image
    LONG        bV4XPelsPerMeter ;  // horizontal resolution
    LONG        bV4YPelsPerMeter ;  // vertical resolution
    DWORD       bV4ClrUsed ;        // number of colors used
    DWORD       bV4ClrImportant ;   // number of important colors
```

```
    DWORD           bV4RedMask ;        // Red color mask
    DWORD           bV4GreenMask ;      // Green color mask
    DWORD           bV4BlueMask ;       // Blue color mask
    DWORD           bV4AlphaMask ;      // Alpha mask
    DWORD           bV4CSType ;         // color space type
    CIEXYZTRIPLE    bV4Endpoints ;      // XYZ values
    DWORD           bV4GammaRed ;       // Red gamma value
    DWORD           bV4GammaGreen ;     // Green gamma value
    DWORD           bV4GammaBlue ;      // Blue gamma value
}
BITMAPV4HEADER, * PBITMAPV4HEADER ;
```

注意:前 11 个字段与 BITMAPINFOHEADER 结构中的相同,后 5 个字段支持 Windows 95 和 Windows NT 4.0 的图像颜色匹配技术。除非使用 BITMAPV4HEADER 结构的后 4 个字段,否则应该使用 BITMAPINFOHEADER(或 BITMAPV5HEADER)。

当 bV4Compression 字段等于 BI_BITFIELDS 时,bV4RedMask、bV4GreenMask 和 bV4BlueMask 可以用于 16 位和 32 位 DIB。它们作为定义在 BITMAPINFOHEADER 结构中的颜色掩码用于相同的函数,并且当使用除了明确的结构字段之外的原始结构时,它们实际上出现在 DIB 文件的相同位置。就我所知,不使用 bV4AlphaMask 字段。

BITMAPV5HEADER 结构剩余的字段包括"Windows 图像颜色管理(Windows Image Color Management)",它的内容超越了本书的范围,但是了解一些背景会对您有益。

为色彩使用 RGB 方案的问题在于,它依赖于视频显示器、彩色照相机和彩色扫描器的技术。如果颜色指定为 RGB 值(255,0,0),意味着最大的电压应该应用到阴极射线管内的红色电子枪上,RGB 值(128,0,0)表示使用一半电压。显示器能以不同的方式响应。而且,打印机使用了不同的颜色方法,包括青色、品红色、黄色和黑色的组合。这些方法称之为"CMY(cyan-magenta-yellow:青色-品红色-黄色)"和"CMYK(cyan-magenta-yellow-black:青色-品红色-黄色-黑色)"。数学公式能把 RGB 值转化为 CMY 和 CMYK,但不能保证打印机颜色与显示器颜色相匹配。"图像颜色管理"是把颜色与对设备无关的标准联系起来的一种尝试。

颜色的现象与可见光的波长有关,波长的范围从 380 nm(蓝)到 780 nm(红)之间。我们所能察觉的全部光线是可见光谱内不同波长的组合。1931 年,Commission Internationale de L'Éclairage (International Commission on Illumination,国际照明委员会)或 CIE 开发了一种科学度量颜色的方法。它包括使用三个颜色匹配的函数(名称为 $\bar{x}$、$\bar{y}$ 和 $\bar{z}$),它们以其省略的形式(带有每 5 nm 的值)发表在 CIE Publication 15.2-1986,"Colorimetry, Second Edition"的表 2.1 中。

颜色的光谱(S)是一组指出每个波长强度的值。如果知道光谱,就能够将与颜色相关的函数应用到光谱来计算 X、Y 和 Z:

$$X = \sum_{\lambda=380}^{780} S(\lambda)\bar{x}(\lambda)$$

$$Y = \sum_{\lambda=380}^{780} S(\lambda)\bar{y}(\lambda)$$

$$Z = \sum_{\lambda=380}^{780} S(\lambda)\bar{z}(\lambda)$$

这些值称为大 X、大 Y 和大 Z。$\bar{y}$ 颜色匹配函数等于肉眼对可见光谱范围内的光线的反应(它看上去像一条在 380 nm 和 780 nm 点趋近于 0 的钟形曲线)。Y 称为“CIE 亮度”,因为它指出了光线的总体强度。

如果使用 BITMAPV5HEADER 结构,bV4CSType 字段就必须设置为 LCS_CALIBRATED_RGB,值为 0。后 4 个字节必须设置为有效值。

CIEXYZTRIPLE 结构定义如下:

```
typedef struct tagCIEXYZTRIPLE
{
    CIEXYZ    ciexyzRed;
    CIEXYZ    ciexyzGreen;
    CIEXYZ    ciexyzBlue;
}
CIEXYZTRIPLE, * LPCIEXYZTRIPLE;
```

而 CIEXYZ 结构定义如下:

```
typedef struct tagCIEXYZ
{
    FXPT2DOT30 ciexyzX;
    FXPT2DOT30 ciexyzY;
    FXPT2DOT30 ciexyzZ;
}
CIEXYZ, * LPCIEXYZ;
```

这三个字段定义为 FXPT2DOT30 值,这意味着它们是带有 2 位整数部分和 30 位小数部分的定点值。这样,0x40000000 是 1.0,0x48000000 是 1.5。最大值 0xFFFFFFFF 仅比 4.0 小一点点。

bV4Endpoints 字段提供了三个与 RGB 颜色(255, 0, 0)、(0, 255, 0)和(0, 0, 255)相关的 X、Y 和 Z 值。这些值应该由创建 DIB 的应用程序插入,以指明这些 RGB 颜色的设备无关的意义。

BITMAPV4HEADER 剩余的三个字段指“伽玛”(希腊的小写字母 γ),它指的是颜色级别的非线性关系。在 DIB 内,红、绿和蓝的范围从 0 到 225。在显示卡上,这三个数值被转化为显示器使用的三个模拟电压,电压决定了每个像素的强度。然而,由于阴极射线管中电子枪的电子特性,像素的强度(I)并不与电压(V)线性相关,它们的关系为

$$I = (V + \varepsilon)^{\gamma}$$

其中 ε 是由显示器的“亮度”控制设置的黑色级别(理想值为 0)。指数 γ 由显示器的

“图像”或“对比度”控制设置的。对于大多数显示器，γ 大约在 2.5 左右。

为了对这种非线性做出补偿，视频摄像机一般会在其线路内包含了“伽玛值修正”。指数 0.45 修正了进入摄像机的光线，这意味着视频显示器的伽玛值为 2.2。（视频显示器的高伽玛值增加了对比度，这通常是没有必要的，因为周围的光线更适合于低对比度。）

视频显示器的这个非线性反应实际上是很适当的，这是因为人类对光线的反应也是非线性的。我曾提到过，Y 被称为“CIE 亮度”，这是线性的光线度量。CIE 还定义了一个接近于人类感觉的亮度值。该感觉亮度用 L^*（发音为“ell star”）表示，它可通过使用如下公式从 Y 计算得到：

$$L^* = \begin{cases} 903.3\dfrac{Y}{Y_n}, & \dfrac{Y}{Y_n} \leqslant 0.008856 \\ 116\left(\dfrac{Y}{Y_n}\right)^{\frac{1}{3}} - 16, & \dfrac{Y}{Y_n} > 0.008856 \end{cases}$$

此处 Y_n 是白色级别。公式的第一部分是一个小的线性部分。通常情况下，人类的亮度感觉是与线性亮度的立方根相关的，这由第二个公式指出。L^* 的范围为 0 ~ 100，L^* 的每个整数增量都被假定是人类能感觉到的亮度的最小变化。

根据感觉亮度而不是线性亮度对光线强度编码要更好一些。这使得位的数量减少到一个合理的程度，并且在模拟线路上也减少了噪音。

让我们来看一下整个过程。像素值（P）范围从 0 到 255，它被线性转化成电压级别，我们假定标准化为 0.0 ~ 1.0 之间的值。假定显示器的黑色级设置为 0，则像素的强度为

$$I = V^{\gamma} = \left(\frac{P}{255}\right)^{\gamma}$$

这里 γ 约为 2.5。人类感觉的亮度（L^*）依赖于此强度的立方根，以及 0 ~ 100 的变化范围，因此约为

$$L^* = 100\left(\frac{P}{255}\right)^{\frac{\gamma}{3}}$$

指数值大约为 0.85。如果指数值为 1，那么 CIE 亮度与像素值完全匹配。当然不完全是那种情况，但是如果像素值指出了线性亮度就非常接近。

BITMAPV4HEADER 的最后三个字段为创建 DIB 的程序提供了一种为像素值指出假设的伽玛值的方法。这些值被解释为 16 位整数值和 16 位的小数值。例如，0x10000 为 1.0。如果 DIB 是通过捕获现实世界的图像而创建的，那么捕获硬件就可能包含这个伽玛值，并且可能是 2.2（编码为 0x23333）。如果 DIB 是由程序通过计算得来的，那么该程序会使用一个强大的函数将它使用的任何线性亮度转化为 CIE 亮度值。该指数的倒数就是在 DIB 中编码的伽玛值。

15.1.9 第 5 版的 Header

为 Windows 98 和 Windows NT 5.0 编写的程序可以使用拥有新的 BITMAPV5HEADER 信息结构的 DIB：

```
typedef struct
{
DWORD        bV5Size ;          // size of the structure = 120
LONG         bV5Width ;         // width of the image in pixels
LONG         bV5Height ;        // height of the image in pixels
WORD         bV5Planes ;        // = 1
WORD         bV5BitCount ;      // bits per pixel (1, 4, 8, 16, 24, or 32)
DWORD        bV5Compression ;   // compression code
DWORD        bV5SizeImage ;     // number of bytes in image
LONG         bV5XPelsPerMeter ; // horizontal resolution
LONG         bV5YPelsPerMeter ; // vertical resolution
DWORD        bV5ClrUsed ;       // number of colors used
DWORD        bV5ClrImportant ;  // number of important colors
DWORD        bV5RedMask ;       // Red color mask
DWORD        bV5GreenMask ;     // Green color mask
DWORD        bV5BlueMask ;      // Blue color mask
DWORD        bV5AlphaMask ;     // Alpha mask
DWORD        bV5CSType ;        // color space type
CIEXYZTRIPLE bV5Endpoints ;     // XYZ values
DWORD        bV5GammaRed ;      // Red gamma value
DWORD        bV5GammaGreen ;    // Green gamma value
DWORD        bV5GammaBlue ;     // Blue gamma value
DWORD        bV5Intent ;        // rendering intent
DWORD        bV5ProfileData ;   // profile data or filename
DWORD        bV5ProfileSize ;   // size of embedded data or filename
DWORD        bV5Reserved ;
}
BITMAPV5HEADER, * PBITMAPV5HEADER ;
```

这里有 4 个新字段，其中只有 3 个有用。这些字段支持 ICC Profile Format Specification，这是由“国际色彩协会(International Color Consortium)”(由 Adobe、Agfa、Apple、Kodak、Microsoft、Silicon Graphics、Sun Microsystems，以及其他公司组成)建立的。您可以从 http://www.icc.org 获取这个标准的副本。基本上，每个输入设备(扫描仪和摄像机)、输出设备(打印机和胶片记录器)，以及显示(监视)设备都与将原始设备有关颜色(一般为 RGB 或 CMYK)联系到设备无关颜色规格的配置文件有关，最终基于 CIE XYZ 值。这些配置文件的扩展名是.ICM(指“图像颜色管理(image color management)”)。配置文件可以嵌入到 DIB 文件中，或从 DIB 文件链接，以指出创建 DIB 的方式。您可以从/Platform SDK/Graphics and Multimedia Services/Color Management 获取有关 Windows“图像颜色管

理"的详细信息。

BITMAPV5HEADER 的 bV5CSType 字段可以拥有几个不同的值。如果是 LCS_CALIBRATED_RGB,那么它就与 BITMAPV4HEADER 结构兼容。bV5Endpoints 字段和伽玛字段必须有效。

如果 bV5CSType 字段是 LCS_sRGB,就不用设置剩余的字段。隐含的颜色空间是"标准"的 RGB 颜色空间,这是由 Microsoft 和 Hewlett-Packard 主要为 Internet 设计的,它包含与设备无关的内容,而不需要大量的配置文件。此文档位于 http://www.color.org/contrib/sRGB.html。

如果 bV5CSType 字段是 LCS_Windows_COLOR_SPACE,就不用设置剩余的字段。Windows 通过 API 函数调用,使用隐含的颜色空间来显示位图。

如果 bV5CSType 字段是 PROFILE_EMBEDDED,则 DIB 文件包含一个 ICC 配置文件。如果字段是 PROFILE_LINKED,DIB 文件就包含 ICC 配置文件的完整路径和文件名。在这两种情况下,bV5ProfileData 都是从 BITMAPV5HEADER 开始到配置文件数据或文件名起始的偏移量。bV5ProfileSize 字段给出了数据或文件名的大小。不必设置结束点和伽玛字段。

15.1.10　显示 DIB 信息

现在我们来看一些代码。实际上我们并不充分了解显示 DIB 的知识,但我们至少能从头结构上显示有关 DIB 的信息。如程序 15-1 DIBHEADS 所示。

程序 15-1　DIBHEADS 程序

DIBHEADS.C

```
/*-----------------------
   DIBHEADS.C——Displays DIB Header Information
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("DibHeads") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
```

```
     HACCEL      hAccel ;
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("DIB Headers"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          if (!TranslateAccelerator (hwnd, hAccel, &msg))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     }
     return msg.wParam ;
}

void Printf (HWND hwnd, TCHAR * szFormat, ...)
{
```

```
     TCHAR    szBuffer [1024] ;
     va_list pArgList ;

     va_start (pArgList, szFormat) ;
     wvsprintf (szBuffer, szFormat, pArgList) ;
     va_end (pArgList) ;

     SendMessage (hwnd, EM_SETSEL, (WPARAM) -1, (LPARAM) -1) ;
     SendMessage (hwnd, EM_REPLACESEL, FALSE, (LPARAM) szBuffer) ;
     SendMessage (hwnd, EM_SCROLLCARET, 0, 0) ;
}

void DisplayDibHeaders (HWND hwnd, TCHAR * szFileName)
{
     static TCHAR          * szInfoName [] = { TEXT ("BITMAPCOREHEADER"),
                                               TEXT ("BITMAPINFOHEADER"),
                                               TEXT ("BITMAPV4HEADER"),
                                               TEXT ("BITMAPV5HEADER") } ;
     static TCHAR          * szCompression [] = { TEXT ("BI_RGB"), TEXT ("BI_RLE8"),
                                                  TEXT ("BI_RLE4"),
                                                  TEXT ("BI_BITFIELDS"),
                                                  TEXT ("unknown") } ;
     BITMAPCOREHEADER * pbmch ;
     BITMAPFILEHEADER * pbmfh ;
     BITMAPV5HEADER   * pbmih ;
     BOOL               bSuccess ;
     DWORD              dwFileSize, dwHighSize, dwBytesRead ;
     HANDLE             hFile ;
     int                i ;
     PBYTE              pFile ;
     TCHAR            * szV ;
          // Display the file name

     Printf (hwnd, TEXT ("File: %s \r \n \r \n"), szFileName) ;

          // Open the file

     hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                         OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
     if (hFile == INVALID_HANDLE_VALUE)
     {
          Printf (hwnd, TEXT ("Cannot open file. \r \n \r \n")) ;
          return ;
     }

          // Get the size of the file
```

```
dwFileSize = GetFileSize (hFile, &dwHighSize) ;

if (dwHighSize)
{
     Printf (hwnd, TEXT ("Cannot deal with > 4G files. \r\n\r\n")) ;
     CloseHandle (hFile) ;
     return ;
}
    // Allocate memory for the file

pFile = malloc (dwFileSize) ;


if (!pFile)
{
     Printf (hwnd, TEXT ("Cannot allocate memory. \r\n\r\n")) ;
     CloseHandle (hFile) ;
     return ;
}

    // Read the file

SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
ShowCursor (TRUE) ;

bSuccess = ReadFile (hFile, pFile, dwFileSize, &dwBytesRead, NULL) ;

ShowCursor (FALSE) ;
SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

if (!bSuccess || (dwBytesRead != dwFileSize))
{
     Printf (hwnd, TEXT ("Could not read file. \r\n\r\n")) ;
     CloseHandle (hFile) ;
     free (pFile) ;
     return ;
}

    // Close the file

CloseHandle (hFile) ;

    // Display file size

Printf (hwnd, TEXT ("File size = %u bytes \r\n\r\n"), dwFileSize) ;

    // Display BITMAPFILEHEADER structure
```

```
pbmfh = (BITMAPFILEHEADER *) pFile ;

Printf (hwnd, TEXT ("BITMAPFILEHEADER \r\n")) ;
Printf (hwnd, TEXT ("\t.bfType = 0x%X \r\n"), pbmfh -> bfType) ;
Printf (hwnd, TEXT ("\t.bfSize = %u \r\n"), pbmfh -> bfSize) ;
Printf (hwnd, TEXT ("\t.bfReserved1 = %u \r\n"), pbmfh -> bfReserved1) ;
Printf (hwnd, TEXT ("\t.bfReserved2 = %u \r\n"), pbmfh -> bfReserved2) ;
Printf (hwnd, TEXT ("\t.bfOffBits = %u \r\n\r\n"), pbmfh -> bfOffBits) ;

    // Determine which information structure we have

pbmih = (BITMAPV5HEADER *) (pFile + sizeof (BITMAPFILEHEADER)) ;

switch (pbmih -> bV5Size)
{
case sizeof (BITMAPCOREHEADER):  i = 0 ;                          break ;
case sizeof (BITMAPINFOHEADER):  i = 1 ;  szV = TEXT ("i") ;      break ;
case sizeof (BITMAPV4HEADER):    i = 2 ;  szV = TEXT ("V4") ;     break ;
case sizeof (BITMAPV5HEADER):    i = 3 ;  szV = TEXT ("V5") ;     break ;
default:
    Printf (hwnd, TEXT ("Unknown header size of %u. \r\n\r\n"),
            pbmih -> bV5Size) ;
    free (pFile) ;
    return ;
}

Printf (hwnd, TEXT ("%s \r\n"), szInfoName[i]) ;

    // Display the BITMAPCOREHEADER fields

if (pbmih -> bV5Size == sizeof (BITMAPCOREHEADER))
{
    pbmch = (BITMAPCOREHEADER *) pbmih ;

    Printf (hwnd, TEXT ("\t.bcSize = %u \r\n"), pbmch -> bcSize) ;
    Printf (hwnd, TEXT ("\t.bcWidth = %u \r\n"), pbmch -> bcWidth) ;
    Printf (hwnd, TEXT ("\t.bcHeight = %u \r\n"), pbmch -> bcHeight) ;
    Printf (hwnd, TEXT ("\t.bcPlanes = %u \r\n"), pbmch -> bcPlanes) ;
    Printf (hwnd, TEXT ("\t.bcBitCount = %u \r\n\r\n"), pbmch -> bcBitCount) ;
    free (pFile) ;
    return ;
}

    // Display the BITMAPINFOHEADER fields

Printf (hwnd, TEXT ("\t.b%sSize = %u \r\n"), szV, pbmih -> bV5Size) ;
Printf (hwnd, TEXT ("\t.b%sWidth = %i \r\n"), szV, pbmih -> bV5Width) ;
Printf (hwnd, TEXT ("\t.b%sHeight = %i \r\n"), szV, pbmih -> bV5Height) ;
Printf (hwnd, TEXT ("\t.b%sPlanes = %u \r\n"), szV, pbmih -> bV5Planes) ;
```

```
Printf (hwnd, TEXT (" \t.b%sBitCount = %u \r \n"), szV, pbmih -> bV5BitCount) ;
Printf (hwnd, TEXT (" \t.b%sCompression = %s \r \n"), szV,
          szCompression [min (4, pbmih -> bV5Compression)]) ;
Printf (hwnd, TEXT (" \t.b%sSizeImage = %u \r \n"), szV, pbmih -> bV5SizeImage) ;
Printf (hwnd, TEXT (" \t.b%sXPelsPerMeter = %i \r \n"), szV,
          pbmih -> bV5XPelsPerMeter) ;
Printf (hwnd, TEXT (" \t.b%sYPelsPerMeter = %i \r \n"), szV,
          pbmih -> bV5YPelsPerMeter) ;
Printf (hwnd, TEXT (" \t.b%sClrUsed = %i \r \n"), szV, pbmih -> bV5ClrUsed) ;
Printf (hwnd, TEXT (" \t.b%sClrImportant = %i \r \n \r \n"), szV,
          pbmih -> bV5ClrImportant) ;

if (pbmih -> bV5Size == sizeof (BITMAPINFOHEADER))
{
     if (pbmih -> bV5Compression == BI_BITFIELDS)
     {
          Printf (hwnd, TEXT ("Red Mask = %08X \r \n"),
                    pbmih -> bV5RedMask) ;
          Printf (hwnd, TEXT ("Green Mask = %08X \r \n"),
                    pbmih -> bV5GreenMask) ;
          Printf (hwnd, TEXT ("Blue Mask = %08X \r \n \r \n"),
                    pbmih -> bV5BlueMask) ;
     }
     free (pFile) ;
     return ;
}

    // Display additional BITMAPV4HEADER fields

Printf (hwnd, TEXT (" \t.b%sRedMask   = %08X \r \n"), szV,
          pbmih -> bV5RedMask) ;
Printf (hwnd, TEXT (" \t.b%sGreenMask = %08X \r \n"), szV,
          pbmih -> bV5GreenMask) ;
Printf (hwnd, TEXT (" \t.b%sBlueMask  = %08X \r \n"), szV,
          pbmih -> bV5BlueMask) ;
Printf (hwnd, TEXT (" \t.b%sAlphaMask = %08X \r \n"), szV,
          pbmih -> bV5AlphaMask) ;
Printf (hwnd, TEXT (" \t.b%sCSType = %u \r \n"), szV,
          pbmih -> bV5CSType) ;
Printf (hwnd, TEXT (" \t.b%sEndpoints.ciexyzRed.ciexyzX   = %08X \r \n"),
          szV, pbmih -> bV5Endpoints.ciexyzRed.ciexyzX) ;
Printf (hwnd, TEXT (" \t.b%sEndpoints.ciexyzRed.ciexyzY   = %08X \r \n"),
          szV, pbmih -> bV5Endpoints.ciexyzRed.ciexyzY) ;
Printf (hwnd, TEXT (" \t.b%sEndpoints.ciexyzRed.ciexyzZ   = %08X \r \n"),
          szV, pbmih -> bV5Endpoints.ciexyzRed.ciexyzZ) ;
Printf (hwnd, TEXT (" \t.b%sEndpoints.ciexyzGreen.ciexyzX = %08X \r \n"),
          szV, pbmih -> bV5Endpoints.ciexyzGreen.ciexyzX) ;
Printf (hwnd, TEXT (" \t.b%sEndpoints.ciexyzGreen.ciexyzY = %08X \r \n"),
```

```
                szV, pbmih -> bV5Endpoints.ciexyzGreen.ciexyzY) ;
     Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzGreen.ciexyzZ = %08X\r\n"),
                szV, pbmih -> bV5Endpoints.ciexyzGreen.ciexyzZ) ;
     Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzBlue.ciexyzX  = %08X\r\n"),
                szV, pbmih -> bV5Endpoints.ciexyzBlue.ciexyzX) ;
     Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzBlue.ciexyzY  = %08X\r\n"),
                szV, pbmih -> bV5Endpoints.ciexyzBlue.ciexyzY) ;
     Printf (hwnd, TEXT ("\t.b%sEndpoints.ciexyzBlue.ciexyzZ  = %08X\r\n"),
                szV, pbmih -> bV5Endpoints.ciexyzBlue.ciexyzZ) ;
     Printf (hwnd, TEXT ("\t.b%sGammaRed    = %08X\r\n"), szV,
                pbmih -> bV5GammaRed) ;
     Printf (hwnd, TEXT ("\t.b%sGammaGreen = %08X\r\n"), szV,
                pbmih -> bV5GammaGreen) ;
     Printf (hwnd, TEXT ("\t.b%sGammaBlue   = %08X\r\n\r\n"), szV,
                pbmih -> bV5GammaBlue) ;
     if (pbmih -> bV5Size == sizeof (BITMAPV4HEADER))
     {
          free (pFile) ;
          return ;
     }

          // Display additional BITMAPV5HEADER fields

     Printf (hwnd, TEXT ("\t.b%sIntent = %u\r\n"), szV, pbmih -> bV5Intent) ;
     Printf (hwnd, TEXT ("\t.b%sProfileData = %u\r\n"), szV,
                pbmih -> bV5ProfileData) ;
     Printf (hwnd, TEXT ("\t.b%sProfileSize = %u\r\n"), szV,
                pbmih -> bV5ProfileSize) ;
     Printf (hwnd, TEXT ("\t.b%sReserved = %u\r\n\r\n"), szV,
                pbmih -> bV5Reserved) ;

     free (pFile) ;
     return ;
}

LRESULT CALLBACK WndProc  (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HWND           hwndEdit ;
     static OPENFILENAME   ofn ;
     static TCHAR          szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR          szFilter[] = TEXT ("Bitmap Files (*.BMP)\0*.bmp\0")
                                        TEXT ("All Files (*.*)\0*.*\0\0") ;

     switch (message)
     {
     case WM_CREATE:
          hwndEdit = CreateWindow (TEXT ("edit"), NULL,
                         WS_CHILD | WS_VISIBLE | WS_BORDER |
```

```
                    WS_VSCROLL | WS_HSCROLL |
                    ES_MULTILINE | ES_AUTOVSCROLL | ES_READONLY,
                0, 0, 0, 0, hwnd, (HMENU) 1,
                ((LPCREATESTRUCT) lParam) -> hInstance, NULL) ;

    ofn.lStructSize          = sizeof (OPENFILENAME) ;
    ofn.hwndOwner            = hwnd ;
    ofn.hInstance            = NULL ;
    ofn.lpstrFilter          = szFilter ;
    ofn.lpstrCustomFilter    = NULL ;
    ofn.nMaxCustFilter       = 0 ;
    ofn.nFilterIndex         = 0 ;
    ofn.lpstrFile            = szFileName ;
    ofn.nMaxFile             = MAX_PATH ;
    ofn.lpstrFileTitle       = szTitleName ;
    ofn.nMaxFileTitle        = MAX_PATH ;
    ofn.lpstrInitialDir      = NULL ;
    ofn.lpstrTitle           = NULL ;
    ofn.Flags                = 0 ;
    ofn.nFileOffset          = 0 ;
    ofn.nFileExtension       = 0 ;
    ofn.lpstrDefExt          = TEXT ("bmp") ;
    ofn.lCustData            = 0 ;
    ofn.lpfnHook             = NULL ;
    ofn.lpTemplateName       = NULL ;
    return 0 ;

case WM_SIZE:
    MoveWindow (hwndEdit, 0, 0, LOWORD (lParam), HIWORD (lParam), TRUE) ;
    return 0 ;

case WM_COMMAND:
    switch (LOWORD (wParam))
    {
    case IDM_FILE_OPEN:
        if (GetOpenFileName (&ofn))
            DisplayDibHeaders (hwndEdit, szFileName) ;

        return 0 ;
    }
    break ;

case WM_DESTROY:
    PostQuitMessage (0) ;
    return 0 ;
}
return DefWindowProc (hwnd, message, wParam, lParam) ;
```

```
}
```

DIBHEADS.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Accelerator
DIBHEADS ACCELERATORS DISCARDABLE
BEGIN
    "O",                IDM_FILE_OPEN,          VIRTKEY, CONTROL, NOINVERT
END

//////////////////////////////////////////////////////////////////////////////
// Menu
DIBHEADS MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open \ tCtrl+O",                 IDM_FILE_OPEN
    END
END
```

RESOURCE.H(摘录)

```
// Microsoft Developer Studio generated include file.
// Used by DibHeads.rc

#define IDM_FILE_OPEN                40001
```

此程序有一个短的 WndProc 函数，它创建了一个只读的编辑窗口充满它的客户区，它也从菜单上处理 File Open 命令。通过调用 GetOpenFileName 函数使用标准的 File Open 对话框，然后调用 DisplayDibHeaders 函数。此函数把整个 DIB 文件读入内存，并逐个字段地显示所有的头信息。

15.2　显示和打印

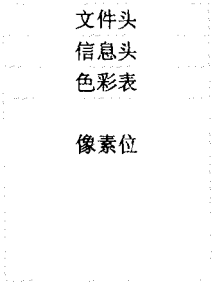
位图是用来看的。在这一节，我们看一看 Windows 支持的用于在视频显示器上或打印页面上显示 DIB 的两个函数。要得到更好的性能，您可以使用一种迂回的方法来显示位图，我会在本章的后面讨论该方法，但先研究这两个函数要好一些。

这两个函数被称为“SetDIBitsToDevice”（发音为“set dee eye bits to device”）和“StretchDIBits”（发音为“stretch dee eye bits”）。每个函数都使用存储在内存中的 DIB，并能显示整个 DIB 或它的矩形部分。当使用 SetDIBitsToDevice 时，以像素为单位显示的图像大小

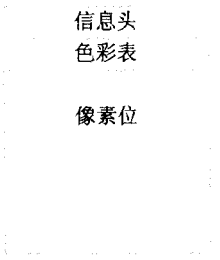
与 DIB 的像素大小相同。例如，一个 640×480 的 DIB 会占据整个标准的 VGA 屏幕，但在300 dpi的激光打印机上它只有约 2.1 英寸×1.6 英寸。StretchDIBits 能缩放 DIB 尺寸的行和列，从而在输出设备上显示一个特定的大小。

15.2.1 进入 DIB

当调用两个函数之一来显示 DIB 时，您需要一些关于图像的信息。正如前面说过的，DIB 文件包含下列部分：



DIB 文件能被加载到内存中。如果除了文件头外，整个文件被存储在连续的内存块中，那么指向该块开始(也就是信息标题的开头)的指针被称作指向紧缩 DIB 的指针(见下图)。



上图所示是通过剪贴板转移 DIB 时所用的格式，并且这也是从 DIB 创建画刷时所用的格式。因为整个 DIB 由单个指针(如 pPackedDib)引用，所以紧缩 DIB 是在内存中存储 DIB 的方便方法，您可以把指针定义为指向 BYTE 的指针。使用本章前面所示的结构定义，能得到所有存储在 DIB 内的信息，包括色彩表和单个的像素位。

然而，要想得到这么多信息，需要若干行代码。例如，您不能通过以下语句简单地访问 DIB 的像素宽度：

```
iWidth = ((PBITMAPINFOHEADER) pPackedDib) -> biWidth ;
```

如果 DIB 为 OS/2 兼容格式，这是可能的。在那种格式中，紧缩的 DIB 以 BITMAPCORE-

HEADER 结构开始，并且 DIB 的像素宽度和高度以 16 位 WORD 而不是 32 位 LONG 存储。因此，必须首先检查 DIB 是否为旧的格式，然后进行相应的操作：

```
if (((PBITMAPCOREHEADER) pPackedDib) -> bcSize == sizeof (BITMAPCOREHEADER))
    iWidth = ((PBITMAPCOREHEADER) pPackedDib) -> bcWidth ;
else
    iWidth = ((PBITMAPINFOHEADER) pPackedDib) -> biWidth ;
```

当然，这不很糟，但它绝不像我们所期望的那样清晰。

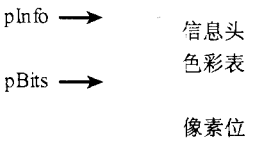
现在有一个很有趣的实验：给定一个指向紧缩 DIB 的指针，发现像素的值位于坐标(5,27)。即使假定 DIB 不是 OS/2 兼容的格式，您也需要了解 DIB 的宽度、高度和位数。您需要计算每一行像素的字节长度，需要确定色彩表内的条目数，以及色彩表是否包括三个 32 位的颜色掩码。还需要检查 DIB 是否被压缩，在这种情况下像素是不能直接由地址得到的。

如果需要直接访问所有的 DIB 像素(许多图像处理工作需要这么做)，这可能会增加一点处理时间。由于这个原因，保存一个指向紧缩 DIB 的指针就很方便了，它本身不是有效的代码。另一个有效的解决方法是为 DIB 定义一个包含足够成员数据的C++类，从而允许快速随机地访问 DIB 像素。然而，我曾答应读者在本书内无需了解C++，所以我将在下一章给出一个 C 的解决方法。

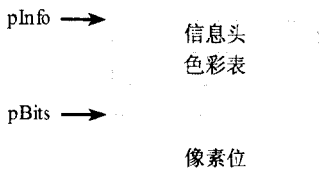
对于 SetDIBitsToDevice 和 StretchDIBits 函数，需要的信息包括一个指向 DIB 的 BITMAPINFO 结构的指针。您应回想起，BITMAPINFO 结构由 BITMAPINFOHEADER 结构和色彩表组成。因此这只是一个指向紧缩 DIB 的指针。

函数也需要一个指向像素位的指针。尽管代码很不漂亮，但还是可以从信息头内的信息得到。注意，当您访问 BITMAPFILEHEADER 结构的 bfOffBits 字段时，这个指针可以很容易地计算出。bfOffBits 字段指出了从 DIB 文件的开头到像素位的偏移量。您可以简单地把此偏移量加到 BITMAPINFO 指针中，然后减去 BITMAPFILEHEADER 结构的大小。然而，当您从剪贴板上得到指向紧缩 DIB 的指针时，这并不起作用，因为没有 BITMAPFILEHEADER 结构。

下图表显示了两个所需的指针：



SetDIBitsToDevice 和 StretchDIBits 函数需要两个指向 DIB 的指针，因为这两部分不在内存的连续块内。您可能有如下所示的两块内存：



把 DIB 分成两个内存块确实是很有用的。这只是因为我们更喜欢与紧缩 DIB 打交道——整个 DIB 是存储在单个内存块中的。

除了这两个指针，SetDIBitsToDevice 和 StretchDIBits 函数通常也需要 DIB 的像素宽度和高度。如果只想显示 DIB 的一部分，就不必明确地知道这些值，但它们会定义您在 DIB 像素位数组内定义的矩形的上限。

15.2.2　像素到像素

SetDIBitsToDevice 函数显示 DIB，并不对其进行拉伸或缩放。DIB 的每个像素映射到输出设备的一个像素上。图像始终被正确地定位——也就是说，图像的顶行在上部。任何会影响设备描述表的转化确定了显示 DIB 是开始位置，但在图像的大小和方向上不受影响。函数如下：

```
iLines = SetDIBitsToDevice (
            hdc,          // device context handle
            xDst,         // x destination coordinate
            yDst,         // y destination coordinate
            cxSrc,        // source rectangle width
            cySrc,        // source rectangle height
            xSrc,         // x source coordinate
            ySrc,         // y source coordinate
            yScan,        // first scan line to draw
            cyScans,      // number of scan lines to draw
            pBits,        // pointer to DIB pixel bits
            pInfo,        // pointer to DIB information
            fClrUse) ;    // color use flag
```

不要对参数的数量感到厌烦。对于大多数用途而言，此函数应比原先出现的更易于使用。而对于另外一些用途，我们会努力让它胜任。

和 GDI 显示函数一样，SetDIBitsToDevice 的第一个参数是设备描述表句柄，它指出

显示 DIB 的设备。下面两个参数 xDst 和 yDst，是输出设备的逻辑坐标，并指出了显示 DIB 图像左上角的坐标（“上端”指的是图像的视觉上端，并不是 DIB 像素的第一行）。注意，这些都是逻辑坐标，因此它们附属于实际上起作用的任何映射方式或（在 Windows NT 的情况下）设置的任何转换。在默认的 MM_TEXT 映射方式下，可以把这些参数设为 0，从显示器表面向左向上显示 DIB 映像。

您可以显示整个 DIB 图像或仅显示其中的一部分，这就是后 4 个参数的作用。但是 DIB 像素数据由上向下的方向产生了许多误解，我一会儿会谈到这些。现在应该清楚，当显示整个 DIB 时，应把 xSrc 和 ySrc 设定为 0，并且 cxSrc 和 cySrc 应分别等于 DIB 的像素宽度和高度。注意，因为 BITMAPINFOHEADER 结构的 biHeight 字段对于从上向下的 DIB 来说是负的，所以 cySrc 应设定为 biHeight 字段的绝对值。

此函数的文档（/Platform SDK/Graphics and Multimedia Services/GDI/Bitmaps/Bitmap Reference/Bitmap Functions/SetDIBitsToDevice）中说，xSrc、ySrc、cxSrc 和 cySrc 参数是逻辑单位。这是不正确的，它们是像素的坐标和尺寸。对于 DIB 内的像素拥有逻辑坐标和单位是没有什么意义的。而且，不管是什么映射方式，在输出设备上显示的 DIB 始终是 cxSrc 像素宽和 cySrc 像素高。

现在对这两个参数 yScan 和 cyScan 不做详细讨论，而仅做简单说明。这些参数在从磁盘文件或通过调制解调器读取数据时，通过每次显示 DIB 的一小部分来减少对内存的需求。通常，yScan 设置为 0，cyScan 设置为 DIB 的高度。

pBits 参数是指向 DIB 像素位的指针。pInfo 参数是指向 DIB 的 BITMAPINFO 结构的指针。虽然 BITMAPINFO 结构的地址与 BITMAPINFOHEADER 结构的地址相同，但是 SetDIBitsToDevice 结构被定义为使用 BITMAPINFO 结构，暗示着：对于 1 位、4 位和 8 位 DIB，位图信息头后必须跟着色彩表。尽管 pInfo 参数被定义为指向 BITMAPINFO 结构的指针，它也是指向 BITMAPCOREINFO、BITMAPV4HEADER 或 BITMAPV5HEADER 结构的指针。

最后一个参数是 DIB_RGB_COLORS 或 DIB_PAL_COLORS，在 WINGDI.H 内分别定义为 0 和 1。如果使用 DIB_RGB_COLORS，这意味着 DIB 包含色彩表。DIB_PAL_COLORS 标志指出，DIB 内的色彩表已经被指向在设备描述表内选定并识别的调色板的 16 位索引代替。在下一章我们学习这个选项。现在使用 DIB_RGB_COLORS，或就是 0。

SetDIBitsToDevice 函数返回所显示的扫描行的数目。

因此，要调用 SetDIBitsToDevice 来显示整个 DIB 图像，您需要下列信息：

- hdc　目标表面的设备描述表句柄；
- xDst 和 yDst　图像左上角的目标坐标；
- cxDib 和 cyDib　DIB 的像素宽度和高度，此处 cyDib 是 BITMAPINFOHEADER 结构内 biHeight 字段的绝对值；

◆ pInfo 和 pBits　指向位图信息部分和像素位的指针。

然后用下列方法调用 SetDIBitsToDevic：

```
SetDIBitsToDevice (
    hdc,          // device context handle
    xDst,         // x destination coordinate
    yDst,         // y destination coordinate
    cxDib,        // source rectangle width
    cyDib,        // source rectangle height
    0,            // x source coordinate
    0,            // y source coordinate
    0,            // first scan line to draw
    cyDib,        // number of scan lines to draw
    pBits,        // pointer to DIB pixel bits
    pInfo,        // pointer to DIB information
    0) ;          // color use flag
```

因此，在 DIB 的 12 个参数中，4 个设定为 0，1 个是重复的。

程序 15-2 SHOWDIB1 通过使用 SetDIBitsToDevice 函数显示 DIB。

程序 15-2　SHOWDIB1 程序

SHOWDIB1.C

```
/*-----------------------
   SHOWDIB1.C——Shows a DIB in the client area
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>
#include "dibfile.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HACCEL     hAccel ;
     HWND       hwnd ;
     MSG        msg ;
     WNDCLASS   wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
```

```
    wndclass.cbClsExtra    = 0 ;
    wndclass.cbWndExtra    = 0 ;
    wndclass.hInstance     = hInstance ;
    wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName  = szAppName ;
    wndclass.lpszClassName = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Show DIB #1"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    hAccel = LoadAccelerators (hInstance, szAppName) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        if (!TranslateAccelerator (hwnd, hAccel, &msg))
        {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BITMAPFILEHEADER * pbmfh ;
    static BITMAPINFO       * pbmi ;
    static BYTE             * pBits ;
    static int                cxClient, cyClient, cxDib, cyDib ;
    static TCHAR              szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
    BOOL                      bSuccess ;
    HDC                       hdc ;
```

```
    PAINTSTRUCT         ps ;

    switch (message)
    {
    case WM_CREATE:
        DibFileInitialize (hwnd) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_INITMENUPOPUP:
        EnableMenuItem ((HMENU) wParam, IDM_FILE_SAVE,
                        pbmfh ? MF_ENABLED : MF_GRAYED) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:
                // Show the File Open dialog box

            if (!DibFileOpenDlg (hwnd, szFileName, szTitleName))
                return 0 ;

                // If there's an existing DIB, free the memory

            if (pbmfh)
            {
                free (pbmfh) ;
                pbmfh = NULL ;
            }
                // Load the entire DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pbmfh = DibLoadImage (szFileName) ;
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                // Invalidate the client area for later update

            InvalidateRect (hwnd, NULL, TRUE) ;
```

```
            if (pbmfh == NULL)
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
                return 0 ;
            }
                // Get pointers to the info structure & the bits

            pbmi = (BITMAPINFO *) (pbmfh + 1) ;
            pBits = (BYTE *) pbmfh + pbmfh -> bfOffBits ;

                // Get the DIB width and height

            if (pbmi -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
            {
                cxDib = ((BITMAPCOREHEADER *) pbmi) -> bcWidth ;
                cyDib = ((BITMAPCOREHEADER *) pbmi) -> bcHeight ;
            }
            else
            {
                cxDib =      pbmi -> bmiHeader.biWidth ;
                cyDib = abs (pbmi -> bmiHeader.biHeight) ;
            }
            return 0 ;

        case IDM_FILE_SAVE:
                // Show the File Save dialog box

            if (!DibFileSaveDlg (hwnd, szFileName, szTitleName))
                return 0 ;

                // Save the DIB to memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            bSuccess = DibSaveImage (szFileName, pbmfh) ;
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (!bSuccess)
                MessageBox (hwnd, TEXT ("Cannot save DIB file"),
                            szAppName, 0) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
```

```
        hdc = BeginPaint (hwnd, &ps) ;

        if (pbmfh)
            SetDIBitsToDevice (hdc,
                               0,              // xDst
                               0,              // yDst
                               cxDib,          // cxSrc
                               cyDib,          // cySrc
                               0,              // xSrc
                               0,              // ySrc
                               0,              // first scan line
                               cyDib,          // number of scan lines
                               pBits,
                               pbmi,
                               DIB_RGB_COLORS) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (pbmfh)
            free (pbmfh) ;

        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIBFILE.H

```
/* - - - - - - - - - - - - - - - - - -
     DIBFILE.H —— Header File for DIBFILE.C
   - - - - - - - - - - - - - - - - - - */
void DibFileInitialize (HWND hwnd) ;
BOOL DibFileOpenDlg (HWND hwnd, PTSTR pstrFileName, PTSTR pstrTitleName) ;
BOOL DibFileSaveDlg (HWND hwnd, PTSTR pstrFileName, PTSTR pstrTitleName) ;

BITMAPFILEHEADER * DibLoadImage (PTSTR pstrFileName) ;
BOOL               DibSaveImage (PTSTR pstrFileName, BITMAPFILEHEADER * ) ;
```

DIBFILE.C

```
/* - - - - - - - - - - - - - - - -
     DIBFILE.C——DIB File Functions
   - - - - - - - - - - - - - - - - */

# include <windows.h>
# include <commdlg.h>
# include "dibfile.h"
```

```
static OPENFILENAME ofn ;

void DibFileInitialize (HWND hwnd)
{
     static TCHAR szFilter[ ] = TEXT ("Bitmap Files (*.BMP) \0*.bmp \0") \
                                TEXT ("All Files (*.*) \0*.* \0\0") ;

     ofn.lStructSize       = sizeof (OPENFILENAME) ;
     ofn.hwndOwner         = hwnd ;
     ofn.hInstance         = NULL ;
     ofn.lpstrFilter       = szFilter ;
     ofn.lpstrCustomFilter = NULL ;
     ofn.nMaxCustFilter    = 0 ;
     ofn.nFilterIndex      = 0 ;
     ofn.lpstrFile         = NULL ;          // Set in Open and Close functions
     ofn.nMaxFile          = MAX_PATH ;
     ofn.lpstrFileTitle    = NULL ;          // Set in Open and Close functions
     ofn.nMaxFileTitle     = MAX_PATH ;
     ofn.lpstrInitialDir   = NULL ;
     ofn.lpstrTitle        = NULL ;
     ofn.Flags             = 0 ;             // Set in Open and Close functions
     ofn.nFileOffset       = 0 ;
     ofn.nFileExtension    = 0 ;
     ofn.lpstrDefExt       = TEXT ("bmp") ;
     ofn.lCustData         = 0 ;
     ofn.lpfnHook          = NULL ;
     ofn.lpTemplateName    = NULL ;
}

BOOL DibFileOpenDlg (HWND hwnd, PTSTR pstrFileName, PTSTR pstrTitleName)
{
     ofn.hwndOwner         = hwnd ;
     ofn.lpstrFile         = pstrFileName ;
     ofn.lpstrFileTitle    = pstrTitleName ;
     ofn.Flags             = 0 ;

     return GetOpenFileName (&ofn) ;
}

BOOL DibFileSaveDlg (HWND hwnd, PTSTR pstrFileName, PTSTR pstrTitleName)
{
     ofn.hwndOwner         = hwnd ;
     ofn.lpstrFile         = pstrFileName ;
     ofn.lpstrFileTitle    = pstrTitleName ;
     ofn.Flags             = OFN_OVERWRITEPROMPT ;

     return GetSaveFileName (&ofn) ;
}
```

```
BITMAPFILEHEADER * DibLoadImage (PTSTR pstrFileName)
{
     BOOL                    bSuccess ;
     DWORD                   dwFileSize, dwHighSize, dwBytesRead ;
     HANDLE                  hFile ;
     BITMAPFILEHEADER *      pbmfh ;

     hFile = CreateFile (pstrFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                         OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
     if (hFile == INVALID_HANDLE_VALUE)
          return NULL ;

     dwFileSize = GetFileSize (hFile, &dwHighSize) ;

     if (dwHighSize)
     {
          CloseHandle (hFile) ;
          return NULL ;
     }

     pbmfh = malloc (dwFileSize) ;
     if (!pbmfh)
     {
          CloseHandle (hFile) ;
          return NULL ;
     }

     bSuccess = ReadFile (hFile, pbmfh, dwFileSize, &dwBytesRead, NULL) ;
     CloseHandle (hFile) ;

     if (!bSuccess || (dwBytesRead != dwFileSize)
                   || (pbmfh -> bfType != * (WORD *) "BM")
                   || (pbmfh -> bfSize != dwFileSize))
     {
          free (pbmfh) ;
          return NULL ;
     }
     return pbmfh ;
}

BOOL DibSaveImage (PTSTR pstrFileName, BITMAPFILEHEADER * pbmfh)
{
     BOOL    bSuccess ;
     DWORD   dwBytesWritten ;
     HANDLE hFile ;

     hFile = CreateFile (pstrFileName, GENERIC_WRITE, 0, NULL,
                         CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL) ;
```

```
    if (hFile == INVALID_HANDLE_VALUE)
        return FALSE ;

    bSuccess = WriteFile (hFile, pbmfh, pbmfh -> bfSize, &dwBytesWritten, NULL) ;
    CloseHandle (hFile) ;
    if (!bSuccess || (dwBytesWritten != pbmfh -> bfSize))
    {
        DeleteFile (pstrFileName) ;
        return FALSE ;
    }
    return TRUE ;
}
```

SHOWDIB1.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
SHOWDIB1 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open...",                    IDM_FILE_OPEN
        MENUITEM "&Save...",                    IDM_FILE_SAVE
    END
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib1.rc

#define IDM_FILE_OPEN                   40001
#define IDM_FILE_SAVE                   40002
```

DIBFILE.C 文件包含显示 File Open 和 File Save 对话框的例程，以及把整个 DIB 文件（拥有 BITMAPFILEHEADER 结构）加载到单个内存块的例程。程序也会将这样一个内存块写入文件。

当在 SHOWDIB1.C 内执行 File Open 命令加载 DIB 文件后，程序计算内存块中 BITMAPINFOHEADER 结构和像素位的偏移量，程序也获得 DIB 的像素宽度和高度。所有信息都存储在静态变量中。在处理 WM_PAINT 消息期间，程序通过调用 SetDIBitsToDevice 来显示 DIB。

当然，SHOWDIB1 还缺少一些功能。例如，如果 DIB 对客户区来说太大，则没有滚动条可用来移动查看。在下一章的末尾将修改这些缺陷。

15.2.3 DIB 的颠倒世界

我们将要学习一个重要的经验，不仅在操作系统的应用程序界面的设计中，而且在生活中。这个经验是：如果一开始就把某些东西弄糟，则当您试图修补它们时，只会把事情弄得更糟。

回到源于 OS/2 的显示管理器的从下向上的 DIB 像素位的定义处，它在一定程度上有意义，因为 PM 内的任何事物都有一个默认的左下角原点。例如，在 PM 窗口内，默认的(0,0)原点是窗口的左下角。(如果您觉得这很古怪，很多人和您的感觉一样。如果您不觉得古怪，那您可能是位数学家。)位图的绘制函数也根据左下角指定目标。

因此，在 OS/2 内如果给位图指定了目标坐标(0,0)，则图像将从窗口的左下角向上向右显示，如图 15-1 所示。

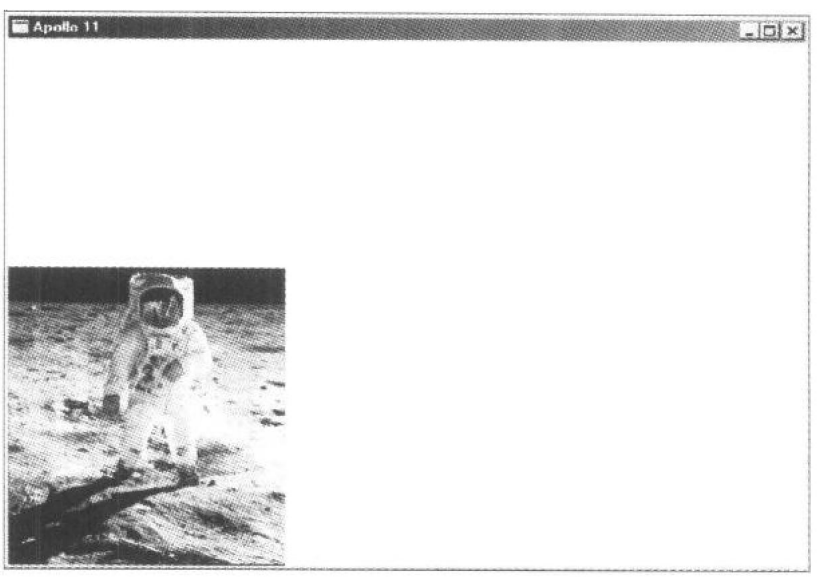


图 15-1　在 OS/2 中以(0,0)目标点显示的位图

用一台足够缓慢的机器，您实际上能看到位图自下向上被绘制的过程。

尽管 OS/2 坐标系统显得很古怪，但它的优点是高度的一致。位图的(0,0)原点是位图文件中第一行的第一个像素，并且此像素被映射到在位图绘制函数中指定的目标坐标上。

Windows 存在的问题是不能保持内部的一致性。当只需要显示整个 DIB 图像中的

一小块矩形时，就要使用参数 xSrc、ySrc、cxSrc 和 cySrc。这些源坐标和大小与 DIB 数据的第一行(图像的最后一行)相关。这方面与 OS/2 相似。然而与 OS/2 不同的是，Windows 在目标坐标上显示图像的顶行。因此，如果显示整个 DIB 图像，显示在(xDst，yDst)的像素是位于坐标(0，cyDib－1)处的像素。DIB 数据的最后一行就是映像的顶行。如果仅显示图像的一部分，则在(xDst，yDst)显示的像素是位于坐标(xSrc，ySrc＋cySrc－1)处的 DIB 像素。

图 15-2 的显示将帮助您理解这方面的内容。下面显示的 DIB 您可以认为它存储在内存中——就是说，上下颠倒。坐标的原点是与 DIB 像素数据的第一个字位一致的。SetDIBitsToDevice 的 xSrc 参数是以 DIB 的左边为基准，并且 cxSrc 是 xSrc 右边的映像宽度，这很直观。ySrc 参数以 DIB 数据的首行(也就是映像的底部)为基准，并且 cySrc 是从 ySrc 到数据的末行(图像的顶端)的映像高度。

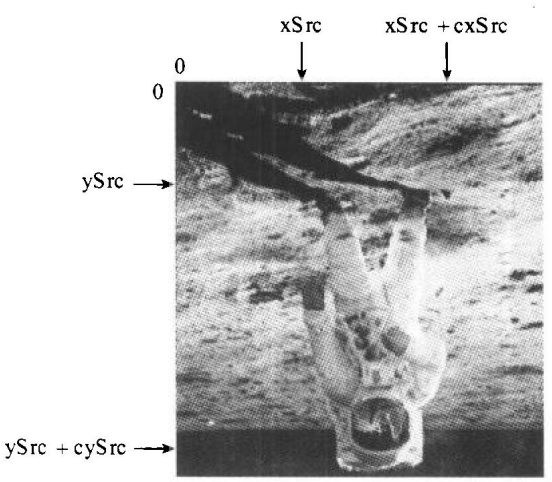


图 15-2　正常 DIB(从下向上)的坐标

如果目标设备描述表具有使用 MM_TEXT 映射方式的默认像素坐标，那么源矩形和目标矩形角上坐标之间的关系如表 15-3 所示。

表 15-3

源矩形	目标矩形
(xSrc，ySrc)	(xDst，yDst＋cySrc－1)
(xSrc＋cxSrc－1，ySrc)	(xDst＋cxSrc－1，yDst＋cySrc－1)
(xSrc，ySrc＋cySrc－1)	(xDst，yDst)
(xSrc＋cxSrc－1，ySrc＋cySrc－1)	(xDst＋cxSrc－1，yDst)

(xSrc，ySrc)不映射到(xDst，yDst)，使表显得很混乱。在其他映射方式中，点(xSrc，

ySrc + cySrc - 1)总是映射到逻辑点(xDst,yDst),图像也与 MM_TEXT 显示的一样。

到目前为止,我们讨论了当 BITMAPINFOHEADER 结构的 biHeight 字段是正值时的正常情况。如果 biHeight 字段是负值,则 DIB 数据以合理的从上向下的方式排列。您可能会认为这样将解决任何问题,如果真地这样认为,那您就错了。

很明显,某人会认为如果把 DIB 上下倒置,旋转每一行,然后给 biHeight 设定一个正值,它将像正常从下向上的 DIB 一样操作,所有与 DIB 矩形相关的现存的代码就不必修改。我认为这是一个合理的目标,但它忘记了一个事实,程序需要修改以处理从上向下的 DIB,这样就不会使用一个负高度。

再者,此决定的结果意味着从上向下的 DIB 的源坐标在 DIB 数据的最后一行有一个原点,它也是图像的底行。这与我们遇到的情况完全不同。位于(0,0)原点的 DIB 像素不再是 pBits 指针引用的第一个像素,它也不是 DIB 文件的最后一个像素,它位于两者之间。

图 15-3 指明了在从上向下的 DIB 中指定矩形的方法,也是它存储在文件或内存中的情景。

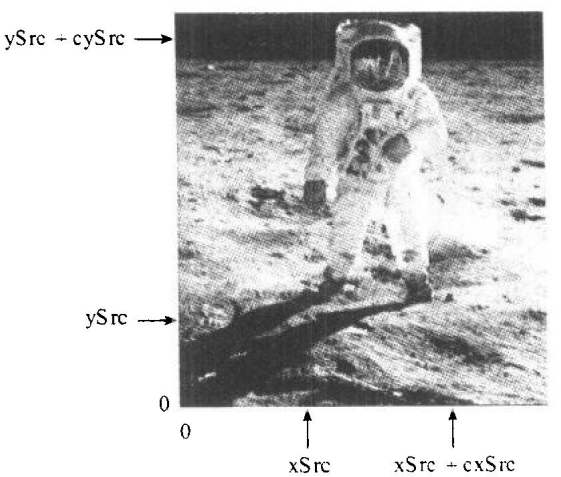


图 15-3　为从上向下的 DIB 指定 DIB 坐标

无论如何,这个方案的实际优点是 SetDIBitsToDevice 函数的参数与 DIB 数据的方向无关。如果有显示了同一图像的两个 DIB(一个从下向上,另一个从上向下。表示在两个 DIB 文件内的行的顺序相反),您可以为 SetDIBitsToDevice 使用相同的参数来选择显示图像的相同部分。

这如程序 15-3 APOLLO11 中所示。

程序 15-3　APOLLO11 程序

APOLLO11.C

```
/*-------------------------
    APOLLO11.C——Program for screen captures
                (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include "dibfile.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("Apollo11") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }
     hwnd = CreateWindow (szAppName, TEXT ("Apollo 11"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
```

```
    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BITMAPFILEHEADER * pbmfh [2] ;
    static BITMAPINFO       * pbmi [2] ;
    static BYTE             * pBits [2] ;
    static int                cxClient, cyClient, cxDib [2], cyDib [2] ;
    HDC                       hdc ;
    PAINTSTRUCT               ps ;

    switch (message)
    {
    case WM_CREATE:
        pbmfh[0] = DibLoadImage (TEXT ("Apollo11.bmp")) ;
        pbmfh[1] = DibLoadImage (TEXT ("ApolloTD.bmp")) ;

        if (pbmfh[0] == NULL || pbmfh[1] == NULL)
        {
            MessageBox (hwnd,TEXT ("Cannot load DIB file"),
                        szAppName, 0) ;
            return 0 ;
        }
        // Get pointers to the info structure & the bits

        pbmi  [0] = (BITMAPINFO *) (pbmfh[0] + 1) ;
        pbmi  [1] = (BITMAPINFO *) (pbmfh[1] + 1) ;
        pBits [0] = (BYTE *) pbmfh[0] + pbmfh[0]->bfOffBits ;
        pBits [1] = (BYTE *) pbmfh[1] + pbmfh[1]->bfOffBits ;

        // Get the DIB width and height (assume BITMAPINFOHEADER)
        // Note that cyDib is the absolute value of the header value!!!

        cxDib [0] =      pbmi[0]->bmiHeader.biWidth ;
        cxDib [1] =      pbmi[1]->bmiHeader.biWidth ;

        cyDib [0] = abs (pbmi[0]->bmiHeader.biHeight) ;
        cyDib [1] = abs (pbmi[1]->bmiHeader.biHeight) ;
        return 0 ;
```

```
case WM_SIZE:
     cxClient = LOWORD (lParam) ;
     cyClient = HIWORD (lParam) ;
     return 0 ;

case WM_PAINT:
     hdc = BeginPaint (hwnd, &ps) ;

          // Bottom-up DIB full size

     SetDIBitsToDevice (hdc,
                        0,                  // xDst
                        cyClient / 4,       // yDst
                        cxDib[0],           // cxSrc
                        cyDib[0],           // cySrc
                        0,                  // xSrc
                        0,                  // ySrc
                        0,                  // first scan line
                        cyDib[0],           // number of scan lines
                        pBits[0],
                        pbmi[0],
                        DIB_RGB_COLORS) ;

          // Bottom-up DIB partial

     SetDIBitsToDevice (hdc,
                        240,                // xDst
                        cyClient / 4,       // yDst
                        80,                 // cxSrc
                        166,                // cySrc
                        80,                 // xSrc
                        60,                 // ySrc
                        0,                  // first scan line
                        cyDib[0],           // number of scan lines
                        pBits[0],
                        pbmi[0],
                        DIB_RGB_COLORS) ;

          // Top-down DIB full size

     SetDIBitsToDevice (hdc,
                        340,                // xDst
                        cyClient / 4,       // yDst
                        cxDib[0],           // cxSrc
                        cyDib[0],           // cySrc
                        0,                  // xSrc
                        0,                  // ySrc
                        0,                  // first scan line
```

```
                    cyDib[0],          // number of scan lines
                    pBits[0],
                    pbmi[0],
                    DIB_RGB_COLORS) ;

          // Top-down DIB partial

          SetDIBitsToDevice (hdc,
                    580,               // xDst
                    cyClient / 4,      // yDst
                    80,                // cxSrc
                    166,               // cySrc
                    80,                // xSrc
                    60,                // ySrc
                    0,                 // first scan line
                    cyDib[1],          // number of scan lines
                    pBits[1],
                    pbmi[1],
                    DIB_RGB_COLORS) ;

          EndPaint (hwnd, &ps) ;
          return 0 ;

     case WM_DESTROY:
          if (pbmfh[0])
               free (pbmfh[0]) ;
          if (pbmfh[1])
               free (pbmfh[1]) ;

          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

程序加载名为 APOLLO11.BMP(从下向上版本)和 APOLLOTD.BMP(从上向下版本)的两个 DIB。它们都是 220 像素宽和 240 像素高。注意:在程序从头信息结构中确定 DIB 的宽度和高度时,它使用 abs 函数得到 biHeight 字段的绝对值。当以全部大小或部分视图显示 DIB 时,无论显示任何位图,xSrc、ySrc、cxSrc 和 cySrc 坐标都是相同的。结果如图 15-4 所示。

注意,"第一条扫描线"和"扫描线数目"参数保持不变,我将在以后简要说明。pBits 参数也不变,不要试图更改 pBits 以使它仅指向需要显示的区域。

我在这个问题上花了这么多时间,并不是因为我要让 Windows 程序员尽力接受 API 的定义中有问题的部分,而是因为如果这个问题引起了混乱,您不至于紧张。它显得混乱是因为它确实被弄乱了。

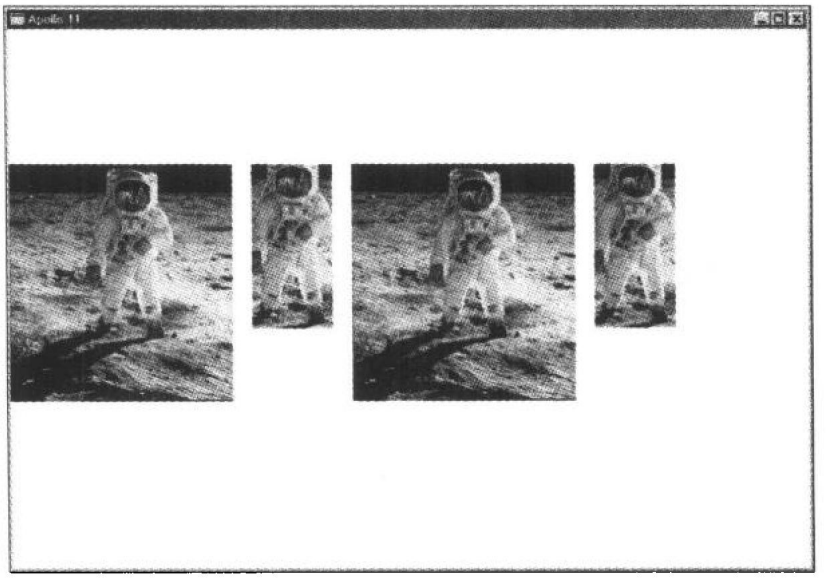


图 15-4　APOLLO11 的显示

我也想让您当心 Windows 文档中的某些语句，例如对于 SetDIBitsToDevice，文档说："自底向上 DIB 的原点是位图的左下角；从上向下 DIB 的原点是左上角"。这不仅模糊，而且是错误的。我可用更好的方式来讲述：从下向上 DIB 的原点是位图图像的左下角，它是位图数据的第一行的第一个像素。从上向下 DIB 的原点也是位图图像的左下角，但在这种情况下，左下角是位图数据的最后一行的第一个像素。

若要编写访问 DIB 的个别位的函数，问题会变糟。这应该与为显示部分 DIB 图像而指定的坐标一致。我的解决方法是（将在第十六章的 DIB 库中实施）同样地引用 DIB 像素和坐标，就像在图像被正确显示时（0,0）原点指 DIB 图像顶行的最左边的像素一样。

15.2.4　连续显示

拥有大量内存能确保程序更容易地运行。显示位于磁盘文件内的 DIB 可以分为两个独立的作业：将 DIB 加载到内存中，然后显示它。

然而，您也可能要在不把整个文件加载到内存中的情况下显示 DIB。即使有足够的物理内存提供给 DIB，把 DIB 移入内存也会迫使 Windows 的虚拟内存系统把内存中别的数据和代码移到磁盘上。如果 DIB 仅用于显示并立即从内存中消除，这就非常讨厌。

还有另一个问题：假定 DIB 位于例如软盘的慢速存储介质上，或由调制解调器传输过来，或者来自从扫描仪或视频帧截取程序取得像素数据的转换例程。您是否等到整

个 DIB 被加载到内存后才显示它？还是当从磁盘或电话线或扫描仪上得到 DIB 时，连续地显示它？

解决这些问题是 SetDIBitsToDevice 函数中 yScan 和 cyScans 参数的目的。要使用这个功能，需要多次调用 SetDIBitsToDevice，大多数情况下使用同样的参数。然而对于每次调用，pBits 参数指向位图像素总体排列的不同部分。yScans 参数指出了 pBits 指向像素数据的行，cyScans 参数是被 pBits 引用的行数。这大量地减少了内存需求。您只需要为保存 DIB 的信息部分（BITMAPINFOHEADER 结构和色彩表）和至少一行像素数据分配足够的内存。

例如，假定 DIB 有 23 行像素，您希望以 5 行的块显示这个 DIB。您可能想分配一个由变量 pInfo 引用的内存块来存储 DIB 的 BITMAPINFO 部分，然后从文件中读取该 DIB。在检查完此结构的字段后，能够计算出一行的字节长度。乘以 5 并分配该大小的另一个块（pBits）。现在读取前 5 行，调用您正常使用的函数，把 yScan 设置为 0，把 cyScans 设置为 5。现在从文件中读取下 5 行，这一次将 yScan 设置为 5，继续将 yScan 设置为 10，然后为 15。最后，将最后 3 行读入 pBits 指向的块，并将 yScan 设置为 20，将 cyScans 设置为 3，调用 SetDIBitsToDevice。

现在有一个不好的消息。首先，使用 SetDIBitsToDevice 的这个功能要求程序的数据获取和数据显示元素之间结合得相当紧密。这通常是不理想的，因为您必须在获得数据和显示数据之间切换。首先，您将延缓整个过程；第二，SetDIBitsToDevice 是具有这个功能的唯一的位图显示函数。StretchDIBits 函数不包括这个功能，因此您不能使用它以不同像素大小显示引入的 DIB。您必须调用 StretchDIBits 多次，每次更改 BITMAPINFOHEADER 结构中的信息，并在屏幕的不同区域显示结果。

程序 15-4 所示的 SEQDISP 演示了如何使用这一特征。

程序 15-4　SEQDISP 程序

SEQDISP.C

```
/*-------------------
   SEQDISP.C——Sequential Display of DIBs
               (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("SeqDisp") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
```

```
                PSTR szCmdLine, int iCmdShow)
{
     HACCEL      hAccel ;
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("DIB Sequential Display"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          if (!TranslateAccelerator (hwnd, hAccel, &msg))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     }
     return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BITMAPINFO * pbmi ;
     static BYTE       * pBits ;
     static int          cxDib, cyDib, cBits ;
     static OPENFILENAME ofn ;
     static TCHAR        szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR        szFilter[] = TEXT ("Bitmap Files (*.BMP) \0*.bmp \0")
                                      TEXT ("All Files (*.*) \0*.* \0\0") ;
     BITMAPFILEHEADER    bmfh ;
     BOOL                bSuccess, bTopDown ;
     DWORD               dwBytesRead ;
     HANDLE              hFile ;
     HDC                 hdc ;
     HMENU               hMenu ;
     int                 iInfoSize, iBitsSize, iRowLength, y ;
     PAINTSTRUCT         ps ;

     switch (message)
     {
     case WM_CREATE:
          ofn.lStructSize       = sizeof (OPENFILENAME) ;
          ofn.hwndOwner         = hwnd ;
          ofn.hInstance         = NULL ;
          ofn.lpstrFilter       = szFilter ;
          ofn.lpstrCustomFilter = NULL ;
          ofn.nMaxCustFilter    = 0 ;
          ofn.nFilterIndex      = 0 ;
          ofn.lpstrFile         = szFileName ;
          ofn.nMaxFile          = MAX_PATH ;
          ofn.lpstrFileTitle    = szTitleName ;
          ofn.nMaxFileTitle     = MAX_PATH ;
          ofn.lpstrInitialDir   = NULL ;
          ofn.lpstrTitle        = NULL ;
          ofn.Flags             = 0 ;
          ofn.nFileOffset       = 0 ;
          ofn.nFileExtension    = 0 ;
          ofn.lpstrDefExt       = TEXT ("bmp") ;
          ofn.lCustData         = 0 ;
          ofn.lpfnHook          = NULL ;
          ofn.lpTemplateName    = NULL ;
          return 0 ;

     case WM_COMMAND:
          hMenu = GetMenu (hwnd) ;

          switch (LOWORD (wParam))
          {
```

```
case IDM_FILE_OPEN:
        // Display File Open dialog
    if (!GetOpenFileName (&ofn))
        return 0 ;

        // Get rid of old DIB

    if (pbmi)
    {
        free (pbmi) ;
        pbmi = NULL ;
    }

    if (pBits)
    {
        free (pBits) ;
        pBits = NULL ;
    }

        // Generate WM_PAINT message to erase background

    InvalidateRect (hwnd, NULL, TRUE) ;
    UpdateWindow (hwnd) ;

    // Open the file

hFile = CreateFile (szFileName, GENERIC_READ,
                    FILE_SHARE_READ, NULL, OPEN_EXISTING,
                    FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;

if (hFile == INVALID_HANDLE_VALUE)
{
    MessageBox (hwnd, TEXT ("Cannot open file."),
                szAppName, MB_ICONWARNING | MB_OK) ;
    return 0 ;
}

    // Read in the BITMAPFILEHEADER

bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                     &dwBytesRead, NULL) ;
if (!bSuccess || dwBytesRead != sizeof (BITMAPFILEHEADER))
{
    MessageBox (hwnd, TEXT ("Cannot read file."),
                szAppName, MB_ICONWARNING | MB_OK) ;
    CloseHandle (hFile) ;
    return 0 ;
```

```
}

    // Check that it's a bitmap

if (bmfh.bfType != * (WORD * ) "BM")
{
     MessageBox (hwnd, TEXT ("File is not a bitmap."),
                 szAppName, MB_ICONWARNING | MB_OK) ;
     CloseHandle (hFile) ;
     return 0 ;
}

    // Allocate memory for header and bits

iInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER) ;
iBitsSize = bmfh.bfSize - bmfh.bfOffBits ;

pbmi = malloc (iInfoSize) ;
pBits = malloc (iBitsSize) ;

if (pbmi == NULL || pBits == NULL)
{
     MessageBox (hwnd, TEXT ("Cannot allocate memory."),
                 szAppName, MB_ICONWARNING | MB_OK) ;
     if (pbmi)
         free (pbmi) ;
     if (pBits)
         free (pBits) ;
     CloseHandle (hFile) ;
     return 0 ;
}

    // Read in the Information Header

bSuccess = ReadFile (hFile, pbmi, iInfoSize,
                     &dwBytesRead, NULL) ;

if (!bSuccess || (int) dwBytesRead != iInfoSize)
{
     MessageBox (hwnd, TEXT ("Cannot read file."),
                 szAppName, MB_ICONWARNING | MB_OK) ;
     if (pbmi)
         free (pbmi) ;
     if (pBits)
         free (pBits) ;
     CloseHandle (hFile) ;
     return 0 ;
}
```

```
        // Get the DIB width and height

    bTopDown = FALSE ;

    if (pbmi -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
    {
        cxDib = ((BITMAPCOREHEADER * ) pbmi) -> bcWidth ;
        cyDib = ((BITMAPCOREHEADER * ) pbmi) -> bcHeight ;
        cBits = ((BITMAPCOREHEADER * ) pbmi) -> bcBitCount ;
    }
    else
    {
        if (pbmi -> bmiHeader.biHeight < 0)
            bTopDown = TRUE ;

        cxDib =        pbmi -> bmiHeader.biWidth ;
        cyDib = abs ( pbmi -> bmiHeader.biHeight) ;
        cBits =        pbmi -> bmiHeader.biBitCount ;

        if (pbmi -> bmiHeader.biCompression != BI_RGB &&
            pbmi -> bmiHeader.biCompression != BI_BITFIELDS)
        {
            MessageBox (hwnd, TEXT ("File is compressed."),
                        szAppName, MB_ICONWARNING | MB_OK) ;
            if (pbmi)
                free (pbmi) ;
            if (pBits)
                free (pBits) ;
            CloseHandle (hFile) ;
            return 0 ;
        }
    }

    // Get the row length

iRowLength = ((cxDib * cBits + 31) & 31) >> 3 ;

    // Read and display

SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
ShowCursor (TRUE) ;

hdc = GetDC (hwnd) ;

for (y = 0 ; y < cyDib ; y++ )
{
    ReadFile (hFile, pBits + y * iRowLength, iRowLength,
              &dwBytesRead, NULL) ;
```

```
            SetDIBitsToDevice (hdc,
                               0,              // xDst
                               0,              // yDst
                               cxDib,          // cxSrc
                               cyDib,          // cySrc
                               0,              // xSrc
                               0,              // ySrc
                               bTopDown ? cyDib - y - 1 : y,
                                               // first scan line
                               1,              // number of scan lines
                               pBits + y * iRowLength,
                               pbmi,
                               DIB_RGB_COLORS) ;
            }
            ReleaseDC (hwnd, hdc) ;
            CloseHandle (hFile) ;
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (pbmi && pBits)
            SetDIBitsToDevice (hdc,
                               0,              // xDst
                               0,              // yDst
                               cxDib,          // cxSrc
                               cyDib,          // cySrc
                               0,              // xSrc
                               0,              // ySrc
                               0,              // first scan line
                               cyDib,          // number of scan lines
                               pBits,
                               pbmi,
                               DIB_RGB_COLORS) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (pbmi)
            free (pbmi) ;

        if (pBits)
            free (pBits) ;
```

```
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SEQDISP.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////
// Accelerator

SEQDISP ACCELERATORS DISCARDABLE
BEGIN
    "O",            IDM_FILE_OPEN,          VIRTKEY, CONTROL, NOINVERT
END

//////////////////////////////////////////////////////////////////////////
// Menu

SEQDISP MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open...\tCtrl+O",            IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by SeqDisp.rc

#define IDM_FILE_OPEN                   40001
```

在处理 File Open 菜单命令期间，在 SEQDISP.C 内的所有文件 I/O 都会发生。在处理 WM_COMMAND 的最后，程序进入读取单行像素并用 SetDIBitsToDevice 显示该行像素的循环。整个 DIB 保存在内存中以便在处理 WM_PAINT 期间也能显示它。

15.2.5 按需放大

SetDIBitsToDevice 按“像素到像素”的方式把 DIB 送到输出设备。这对于打印 DIB 用处不大。打印机的分辨率越高，得到的图像就越小，您最终会得到邮票大小的图像。

要通过缩小或放大 DIB,在输出设备上以特定的大小显示它,可以使用 StretchDIBits:

```
iLines = StretchDIBits (
            hdc,          // device context handle
            xDst,         // x destination coordinate
            yDst,         // y destination coordinate
            cxDst,        // destination rectangle width
            cyDst,        // destination rectangle height
            xSrc,         // x source coordinate
            ySrc,         // y source coordinate
            cxSrc,        // source rectangle width
            cySrc,        // source rectangle height
            pBits,        // pointer to DIB pixel bits
            pInfo,        // pointer to DIB information
            fClrUse,      // color use flag
            dwRop) ;      // raster operation
```

函数参数除了下列三个方面均与 SetDIBitsToDevice 相同:

- 目标坐标包括逻辑宽度(cxDst)和高度(cyDst),以及开始点;
- 不能通过连续显示 DIB 来减少内存需求;
- 最后一个参数是光栅操作,它指出了 DIB 像素与输出设备像素结合的方式,在最后一章将学到这些内容。现在我们将此参数设置为 SRCCOPY。

还有另一个更细微的差别。如果查看 SetDIBitsToDevice 函数的声明,您会发现 cxSrc 和 cySrc 是 DWORD,这是 32 位无符号长整数。在 StretchDIBits 中,cxSrc 和 cySrc(以及 cxDst 和 cyDst)定义为带符号的整数,这意味着它们可以为负数,实际上一会儿就会看到,它们确实能为负数。如想您已经开始检查是否别的参数也可以为负数,就让我声明一下:在两个函数中,xSrc 和 ySrc 均定义为 int,但这是错的,这些值始终是非负数。

DIB 内的源矩形被映射到目标矩形的坐标显示如表 15-4 所示。其右列中的 -1 项是不精确的,因为放大的程度(以及映射方式和其他变换)能产生略微不同的结果。

表 15-4

源矩形	目标矩形
(xSrc, ySrc)	(xDst, yDst + cyDst - 1)
(xSrc + cxSrc - 1, ySrc)	(xDst + cxDst - 1, yDst + cyDst - 1)
(xSrc, ySrc + cySrc - 1)	(xDst, yDst)
(xSrc + cxSrc - 1, ySrc + cySrc - 1)	(xDst + cxDst - 1, yDst)

例如,考虑一个 2×2 的 DIB,这里 StretchDIBits 的 xSrc 和 ySrc 参数均为 0,cxSrc 和 cySrc 均为 2。假定我们显示到的设备描述表具有 MM_TEXT 映射方式并且不进行变

换。如果 xDst 和 yDst 均为 0,cxDst 和 cyDst 均为 4,那么我们将 DIB 放大 2 倍。每个源像素(x、y)将映射到下面所示的 4 个目标像素上:

(0,0) - -> (0,2) and (1,2) and (0,3) and (1,3)
(1,0) - -> (2,2) and (3,2) and (2,3) and (3,3)
(0,1) - -> (0,0) and (1,0) and (0,1) and (1,1)
(1,1) - -> (2,0) and (3,0) and (2,1) and (3,1)

上表正确地指出了目标的角:(0,3)、(3,3)、(0,0)和(3,0)。在其他情况下,坐标可能是个近似值。

目标设备描述表的映射方式对 SetDIBitsToDevice 的影响仅由于 xDst 和 yDst 是逻辑坐标。StretchDIBits 完全受映射方式的影响。例如,如果您设置了 y 值向上增长的一种度量映射方式,DIB 就会颠倒显示。

可以通过把 cyDst 设置为负数来弥补这种情况。实际上,您可以将任何参数的宽度和高度变为负值来水平或垂直翻转 DIB。在 MM_TEXT 映射方式下,如果 cySrc 和 cyDst 符号相反,DIB 会沿着水平轴翻转并颠倒显示。如果 cxSrc 和 cxDst 符号相反,DIB 会沿着垂直轴翻转并显示它的镜面图像。

下面是总结这些内容的几个表达方式,xMM 和 yMM 指出映射方式的方向,如果 x 值向右增长,则 xMM 值为 1;如果 x 值向左增长,则值为 - 1。同样,如果 y 值向下增长,则 yMM 值为 1;如果 y 值向上增长,则值为 - 1。Sign 函数对于正值返回 TURE,对于负值返回 FALSE。

```
if (!Sign (xMM × cxSrc × cxDst))
    DIB is flipped on its vertical axis (mirror image)

if (!Sign (yMM × cySrc × cyDst))
    DIB is flipped on its horizontal axis (upside down)
```

若有疑问,请查阅表 15-4。

程序 15-5 所示的 SHOWDIB2 以实际尺寸显示 DIB,放大至客户区窗口的大小,打印 DIB,以及把 DIB 转移到剪贴板。

程序 15-5　SHOWDIB2 程序

SHOWDIB2.C

```
/*-----------------------
    SHOWDIB2.C——Shows a DIB in the client area
                (c) Charles Petzold, 1998
   -----------------------*/
```

```
#include <windows.h>
#include "dibfile.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib2") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HACCEL      hAccel ;
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #2"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
```

```
            if (!TranslateAccelerator (hwnd, hAccel, &msg))
            {
                TranslateMessage (&msg) ;
                DispatchMessage (&msg) ;
            }
        }
        return msg.wParam ;
}

int ShowDib (HDC hdc, BITMAPINFO * pbmi, BYTE * pBits, int cxDib, int cyDib,
             int cxClient, int cyClient, WORD wShow)
{
        switch (wShow)
        {
        case IDM_SHOW_NORMAL:
            return SetDIBitsToDevice (hdc, 0, 0, cxDib, cyDib, 0, 0,
                                      0, cyDib, pBits, pbmi, DIB_RGB_COLORS) ;

        case IDM_SHOW_CENTER:
            return SetDIBitsToDevice (hdc, (cxClient - cxDib) / 2,
                                           (cyClient - cyDib) / 2,
                                      cxDib, cyDib, 0, 0,
                                      0, cyDib, pBits, pbmi, DIB_RGB_COLORS) ;
        case IDM_SHOW_STRETCH:
            SetStretchBltMode (hdc, COLORONCOLOR) ;

            return StretchDIBits (hdc, 0, 0, cxClient, cyClient,
                                       0, 0, cxDib, cyDib,
                                  pBits, pbmi, DIB_RGB_COLORS, SRCCOPY) ;

        case IDM_SHOW_ISOSTRETCH:
            SetStretchBltMode (hdc, COLORONCOLOR) ;
            SetMapMode (hdc, MM_ISOTROPIC) ;
            SetWindowExtEx (hdc, cxDib, cyDib, NULL) ;
            SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;
            SetWindowOrgEx (hdc, cxDib / 2, cyDib / 2, NULL) ;
            SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;

            return StretchDIBits (hdc, 0, 0, cxDib, cyDib,
                                       0, 0, cxDib, cyDib,
                                  pBits, pbmi, DIB_RGB_COLORS, SRCCOPY) ;
        }
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        static BITMAPFILEHEADER * pbmfh ;
```

```
static BITMAPINFO       * pbmi ;
static BYTE             * pBits ;
static DOCINFO            di = { sizeof (DOCINFO),
                                 TEXT ("ShowDib2: Printing") } ;
static int                cxClient, cyClient, cxDib, cyDib ;
static PRINTDLG           printdlg = { sizeof (PRINTDLG) } ;
static TCHAR              szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
static WORD               wShow = IDM_SHOW_NORMAL ;
BOOL                      bSuccess ;
HDC                       hdc, hdcPrn ;
HGLOBAL                   hGlobal ;
HMENU                     hMenu ;
int                       cxPage, cyPage, iEnable ;
PAINTSTRUCT               ps ;
BYTE                    * pGlobal ;

switch (message)
{
case WM_CREATE:
     DibFileInitialize (hwnd) ;
     return 0 ;

case WM_SIZE:
     cxClient = LOWORD (lParam) ;
     cyClient = HIWORD (lParam) ;
     return 0 ;

case WM_INITMENUPOPUP:
     hMenu = GetMenu (hwnd) ;

     if (pbmfh)
          iEnable = MF_ENABLED ;
     else
          iEnable = MF_GRAYED ;

     EnableMenuItem (hMenu, IDM_FILE_SAVE,     iEnable) ;
     EnableMenuItem (hMenu, IDM_FILE_PRINT,    iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_CUT,      iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_COPY,     iEnable) ;
     EnableMenuItem (hMenu, IDM_EDIT_DELETE,   iEnable) ;
     return 0 ;

case WM_COMMAND:
     hMenu = GetMenu (hwnd) ;

     switch (LOWORD (wParam))
     {
```

```
case IDM_FILE_OPEN:
        // Show the File Open dialog box

    if (!DibFileOpenDlg (hwnd, szFileName, szTitleName))
        return 0 ;

        // If there's an existing DIB, free the memory
    if (pbmfh)
    {
        free (pbmfh) ;
        pbmfh = NULL ;
    }
        // Load the entire DIB into memory

    SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
    ShowCursor (TRUE) ;

    pbmfh = DibLoadImage (szFileName) ;

    ShowCursor (FALSE) ;
    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

        // Invalidate the client area for later update

    InvalidateRect (hwnd, NULL, TRUE) ;

    if (pbmfh == NULL)
    {
        MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                    szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;
    }
        // Get pointers to the info structure & the bits

    pbmi = (BITMAPINFO *) (pbmfh + 1) ;
    pBits = (BYTE *) pbmfh + pbmfh -> bfOffBits ;

        // Get the DIB width and height

    if (pbmi -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
    {
        cxDib = ((BITMAPCOREHEADER *) pbmi) -> bcWidth ;
        cyDib = ((BITMAPCOREHEADER *) pbmi) -> bcHeight ;
    }
    else
    {
        cxDib =       pbmi -> bmiHeader.biWidth ;
        cyDib = abs (pbmi -> bmiHeader.biHeight) ;
```

```
        }
        return 0 ;

    case IDM_FILE_SAVE:
            // Show the File Save dialog box

        if (!DibFileSaveDlg (hwnd, szFileName, szTitleName))
            return 0 ;

            // Save the DIB to a disk file

        SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
        ShowCursor (TRUE) ;

        bSuccess = DibSaveImage (szFileName, pbmfh) ;

        ShowCursor (FALSE) ;
        SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

        if (!bSuccess)
            MessageBox (hwnd, TEXT ("Cannot save DIB file"),
                        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;

    case IDM_FILE_PRINT:
        if (!pbmfh)
            return 0 ;

            // Get printer DC

        printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;
        if (!PrintDlg (&printdlg))
            return 0 ;

        if (NULL == (hdcPrn = printdlg.hDC))
        {
            MessageBox (hwnd, TEXT ("Cannot obtain Printer DC"),
                        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
            return 0 ;
        }
            // Check whether the printer can print bitmaps

        if (! (RC_BITBLT & GetDeviceCaps (hdcPrn, RASTERCAPS)))
        {
            DeleteDC (hdcPrn) ;
            MessageBox (hwnd, TEXT ("Printer cannot print bitmaps"),
                        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
```

```
                return 0 ;
            }
                // Get size of printable area of page

            cxPage = GetDeviceCaps (hdcPrn, HORZRES) ;
            cyPage = GetDeviceCaps (hdcPrn, VERTRES) ;

            bSuccess = FALSE ;
                // Send the DIB to the printer

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
            {
                ShowDib (hdcPrn, pbmi, pBits, cxDib, cyDib,
                         cxPage, cyPage, wShow) ;

                if (EndPage (hdcPrn) > 0)
                {
                    bSuccess = TRUE ;
                    EndDoc (hdcPrn) ;
                }
            }
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            DeleteDC (hdcPrn) ;

            if (!bSuccess)
                MessageBox (hwnd, TEXT ("Could not print bitmap"),
                            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
            return 0 ;

        case IDM_EDIT_COPY:
        case IDM_EDIT_CUT:
            if (!pbmfh)
                return 0 ;

                // Make a copy of the packed DIB

            hGlobal = GlobalAlloc (GHND | GMEM_SHARE, pbmfh -> bfSize -
                                   sizeof (BITMAPFILEHEADER)) ;
            pGlobal = GlobalLock (hGlobal) ;

            CopyMemory (pGlobal, (BYTE *) pbmfh + sizeof (BITMAPFILEHEADER),
                        pbmfh -> bfSize - sizeof (BITMAPFILEHEADER)) ;
            GlobalUnlock (hGlobal) ;
```

```
                // Transfer it to the clipboard

            OpenClipboard (hwnd) ;

            EmptyClipboard () ;
            SetClipboardData (CF_DIB, hGlobal) ;
            CloseClipboard () ;

            if (LOWORD (wParam) == IDM_EDIT_COPY)
                return 0 ;
                                        // fall through if IDM_EDIT_CUT
        case IDM_EDIT_DELETE:
            if (pbmfh)
            {
                free (pbmfh) ;
                pbmfh = NULL ;
                InvalidateRect (hwnd, NULL, TRUE) ;
            }
            return 0 ;

        case IDM_SHOW_NORMAL:
        case IDM_SHOW_CENTER:
        case IDM_SHOW_STRETCH:
        case IDM_SHOW_ISOSTRETCH:
            CheckMenuItem (hMenu, wShow, MF_UNCHECKED) ;
            wShow = LOWORD (wParam) ;
            CheckMenuItem (hMenu, wShow, MF_CHECKED) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (pbmfh)
            ShowDib (hdc, pbmi, pBits, cxDib, cyDib,
                     cxClient, cyClient, wShow) ;

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (pbmfh)
            free (pbmfh) ;
```

```
            PostQuitMessage (0) ;
            return 0 ;
        }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB2.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

SHOWDIB2 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open...\tCtrl+O",            IDM_FILE_OPEN
        MENUITEM "&Save...\tCtrl+S",            IDM_FILE_SAVE
        MENUITEM SEPARATOR
        MENUITEM "&Print\tCtrl+P",              IDM_FILE_PRINT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "Cu &t\tCtrl+X",               IDM_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",               IDM_EDIT_COPY
        MENUITEM "&Delete\tDelete",             IDM_EDIT_DELETE
    END
    POPUP "&Show"
    BEGIN
        MENUITEM "&Actual Size",                IDM_SHOW_NORMAL, CHECKED
        MENUITEM "&Center",                     IDM_SHOW_CENTER
        MENUITEM "&Stretch to Window",          IDM_SHOW_STRETCH
        MENUITEM "Stretch &Isotropically",      IDM_SHOW_ISOSTRETCH
    END
END

//////////////////////////////////////////////////////////////////////////////
// Accelerator

SHOWDIB2 ACCELERATORS DISCARDABLE
BEGIN
    "C",            IDM_EDIT_COPY,      VIRTKEY, CONTROL, NOINVERT
    "O",            IDM_FILE_OPEN,      VIRTKEY, CONTROL, NOINVERT
    "P",            IDM_FILE_PRINT,     VIRTKEY, CONTROL, NOINVERT
```

```
    "S",            IDM_FILE_SAVE,      VIRTKEY, CONTROL, NOINVERT
    VK_DELETE,      IDM_EDIT_DELETE,    VIRTKEY, NOINVERT
    "X",            IDM_EDIT_CUT,       VIRTKEY, CONTROL, NOINVERT
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib2.rc

#define IDM_FILE_OPEN               40001
#define IDM_SHOW_NORMAL             40002
#define IDM_SHOW_CENTER             40003
#define IDM_SHOW_STRETCH            40004
#define IDM_SHOW_ISOSTRETCH         40005
#define IDM_FILE_PRINT              40006
#define IDM_EDIT_COPY               40007
#define IDM_EDIT_CUT                40008
#define IDM_EDIT_DELETE             40009
#define IDM_FILE_SAVE               40010
```

有意思的是 ShowDib 函数,它根据菜单选择以 4 种不同的方式之一在程序的客户区显示 DIB。可以使用 SetDIBitsToDevice 从客户区的左上角或在客户区的中心显示 DIB。程序也有两个使用 StretchDIBits 的选项,DIB 能放大填充整个客户区,在此情况下它可能会变形,或它能等比例显示,也就是说不会变形。

把 DIB 复制到剪贴板包括:在全局共享内存中制作紧缩 DIB 内存块的副本。剪贴板数据类型为 CF_DIB。程序没有演示从剪贴板复制 DIB 的方法,因为这需要更多的逻辑在仅有指向紧缩 DIB 的指针的情况下确定像素位的偏移量。我将在下一章的末尾演示方法。

您可能注意到了 SHOWDIB2 中的一些不足。如果以 256 色视频模式运行 Windows,就会看到显示除了单色或 4 位 DIB 以外的其他图形出现的问题,您看不到真正的颜色。访问那些颜色需要使用调色板,在下一章会做这些工作。您也可能注意到速度问题,尤其在 Windows NT 下运行 SHOWDIB2 时。在下一章紧缩 DIB 和位图时,我会演示处理的方法。我也给 DIB 显示添加滚动条,这样也能以实际尺寸查看大于屏幕的 DIB。

15.2.6　色彩转换、调色板和性能

记得在 William Goldman 的电影剧本《All the President's Men》中,Deep Throat 告诉 Bob Woodward 揭开水门秘密的关键是"跟着钱"。那么在位图显示中获得高级性能的关键是"跟着像素位",以及理解色彩转化发生的时间和位置。DIB 是设备无关的格式,视频显示器内存总是另一种格式。在 SetDIBitsToDevice 或 StretchDIBits 函数调用期间,每个

像素(可能有几百万个)必须从设备无关格式向设备相关格式转换。

在许多情况下,这种转换是很繁琐的。例如,在24位视频显示器上显示24位DIB,显示驱动程序必须以红、绿、蓝的字节顺序进行切换。在24位设备上显示16位DIB需要位的移动和修剪。在24位设备上显示4位或8位DIB要求在DIB色彩表内查找DIB像素位然后对字节重新排列。

但是当要在4位或8位视频显示器上显示16位、24位或32位DIB时,会发生什么呢?一个完全不同类型的颜色转换发生了。对于DIB内的每个像素,设备驱动程序必须在像素和显示器上可用的颜色之间执行"最接近的色彩查找",这包括循环和计算。(GDI函数GetNearestColor执行"最接近的色彩查找"。)

整个RGB色彩的三维数组可用立方体表示。曲线内任意两点之间的距离是

$$\sqrt{(R_2 - R_1)^2 + (G_2 - G_1)^2 + (B_2 - B_1)^2}$$

在这里,两个颜色是$R_1G_1B_1$和$R_2G_2B_2$。执行最接近的色彩查找包括从一种颜色到其他颜色集合中查找最短距离。幸运的是,当在RGB颜色立方体中"比较"距离时,并不需要计算出平方根部分。但是需转换的每个像素必须与设备的所有颜色相比较以发现最接近的颜色。这是个工作量相当大的工作。(尽管在8位设备上显示8位DIB也包括最接近的色彩查找,但它不必对每个像素都进行,它仅需对DIB色彩表中的每种颜色进行查找。)

正是由于以上原因,应该避免使用SetDIBitsToDevice或StretchDIBits在8位视频显示适配器上显示16位、24位或32位DIB。DIB应转化为8位DIB,或者为达到更好的性能,应转化为8位DDB。实际上,能通过将DIB转化为DDB并使用BitBlt和StretchBlt显示图像,来加快显示任何DIB的速度。

如果在8位视频显示器上运行Windows(或仅仅切换到8位模式来观察在显示全色DIB时性能的变化),您可能会注意到另一个问题:DIB不用所有的颜色显示。任何在8位视频显示器上显示的DIB限制在刚好20种颜色。获得多于20种颜色是"调色板管理器"的任务,这将在下一章涉及。

最后,如果在同一台机器上运行Windows 98和Windows NT,可能会注意到:对于同等的视频模式,Windows NT显示大型DIB花费的时间较长。这是Windows NT的客户/服务器体系结构的结果,它是大量数据通过API所造成的不良后果。解决方法是将DIB转化为DDB。而我一会儿谈到的CreateDIBSection函数对这种情况特别有用。

15.3 DIB和DDB的结合

您能够了解关于DIB格式的很多知识并调用两个DIB绘制函数:SetDIBitsToDevice和StretchDIBits。您可以直接访问DIB中的单个位、字节和像素,并且一旦您有一堆能让您以结构的方式检查和更改数据的函数,对您所能做的事就没有限制。

实际上,我们发现还是有一些限制。在上一章,我们了解了使用 GDI 函数在 DDB 上绘制图像的方法。到目前为止,还没有在 DIB 上绘制的方法。另一个问题是 SetDIBitsToDevice 和 StretchDIBits 没有 BitBlt 和 StretchBlt 速度快,尤其在 Windows NT 环境下以及执行许多最接近色彩的查找(例如,在 8 位视频卡上显示 24 位 DIB)时。

因此,在 DIB 和 DDB 之间转换是有益的。例如,如果我们有一个需要在屏幕上显示的 DIB,并且需要显示多次,那么把 DIB 转换为 DDB 就很有意义,这样我们就能够使用快速的 BitBlt 和 StretchBlt 函数。

15.3.1　从 DIB 创建 DDB

从 DIB 中创建 GDI 位图对象可能吗? 基本上我们已知道了方法:如果有 DIB,您可使用 CreateCompatibleBitmap 来创建与 DIB 大小相同并与视频显示器兼容的 GDI 位图对象,然后将该位图对象选入内存设备描述表,并调用 SetDIBitsToDevice 在那个内存 DC 上绘图。结果就是 DDB 具有与 DIB 相同的图像,但具有与视频显示器兼容的颜色结构。

您也可以通过调用 CreateDIBitmap 用几个步骤完成上述工作。函数的语法为:

```
hBitmap = CreateDIBitmap (
              hdc,              // device context handle
              pInfoHdr,         // pointer to DIB information header
              fInit,            // 0 or CBM_INIT
              pBits,            // pointer to DIB pixel bits
              pInfo,            // pointer to DIB information
              fClrUse) ;        // color use flag
```

请注意 pInfoHdr 和 pInfo 这两个参数,它们分别定义为指向 BITMAPINFOHEADER 结构和 BITMAPINFO 结构的指针。正如我们所知,BITMAPINFO 结构是后面紧跟色彩表的 BITMAPINFOHEADER 结构。我们一会儿会看到这种区别所起的作用。最后一个参数是 DIB_RGB_COLORS(等于 0)或 DIB_PAL_COLORS,它们在 SetDIBitsToDevice 函数中使用。下一章我将讨论更多这方面的内容。

理解 Windows 中位图函数的作用是很重要的,不要考虑 CreateDIBitmap 函数的名称,它"不创建与设备无关的位图",它从设备无关的规格中创建"设备相关的位图"。注意该函数返回 GDI 位图对象的句柄,CreateBitmap、CreateBitmapIndirect 和 CreateCompatibleBitmap 也与它一样。

调用 CreateDIBitmap 函数最简单的方法是:

```
hBitmap = CreateDIBitmap (NULL, pbmih, 0, NULL, NULL, 0) ;
```

唯一的参数是指向 BITMAPINFOHEADER 结构(不带色彩表)的指针。在这个形式中,函数创建单色 GDI 位图对象。调用此函数的第二种简单的方法是:

```
hBitmap = CreateDIBitmap (hdc, pbmih, 0, NULL, NULL, 0) ;
```

在这个形式中，函数创建了与设备描述表兼容的 DDB，该设备描述表由 hdc 参数指出。到目前为止，我们都是通过 CreateBitmap（创建单色位图）或 CreateCompatibleBitmap（创建与视频显示器兼容的位图）来完成一些工作。

在 CreateDIBitmap 的这两个简化的模式中，像素还未被初始化。如果 CreateDIBitmap 的第三个参数是 CBM_INIT，Windows 就会创建 DDB，并使用最后三个参数对位图的诸字位进行初始化。pInfo 参数是指向包括色彩表的 BITMAPINFO 结构的指针。pBits 参数是指向由 BITMAPINFO 结构指出的色彩格式中的位数组的指针，根据色彩表这些位被转换为设备的颜色格式，这与 SetDIBitsToDevice 的情况相同。实际上，整个 CreateDIBitmap 函数可以用下列代码来实现：

```
HBITMAP CreateDIBitmap (HDC hdc, CONST BITMAPINFOHEADER * pbmih,
                        DWORD fInit, CONST VOID * pBits,
                        CONST BITMAPINFO * pbmi, UINT fUsage)
{
     HBITMAP  hBitmap ;
     HDC      hdc ;
     int      cx, cy, iBitCount ;

     if (pbmih -> biSize == sizeof (BITMAPCOREHEADER))
     {
          cx          = ((PBITMAPCOREHEADER) pbmih) -> bcWidth ;
          cy          = ((PBITMAPCOREHEADER) pbmih) -> bcHeight ;
          iBitCount   = ((PBITMAPCOREHEADER) pbmih) -> bcBitCount ;
     }
     else
     {
          cx          = pbmih -> biWidth ;
          cy          = pbmih -> biHeight ;
          iBitCount   = pbmih -> biBitCount ;
     }
     if (hdc)
         hBitmap = CreateCompatibleBitmap (hdc, cx, cy) ;
     else
         hBitmap = CreateBitmap (cx, cy, 1, 1, NULL) ;

     if (fInit == CBM_INIT)
     {
          hdcMem = CreateCompatibleDC (hdc) ;
          SelectObject (hdcMem, hBitmap) ;
          SetDIBitsToDevice (hdcMem, 0, 0, cx, cy, 0, 0, 0 cy,
                             pBits, pbmi, fUsage) ;
          DeleteDC (hdcMem) ;
     }
```

```
return hBitmap ;
}
```

如果仅需显示 DIB 一次并对 SetDIBitsToDevice 的性能感到担心，则调用 CreateDIBitmap 并通过使用 BitBlt 或 StretchBlt 来显示 DDB 就没有什么意义。因为 SetDIBitsToDevice 和 CreateDIBitmap 都执行颜色转换，这两个工作会占用同样长的时间。仅当多次显示 DIB(例如在处理 WM_PAINT 消息时)进行这种转换才有意义。

程序 15-6 所示的 DIBCONV 演示了利用 SetDIBitsToDevice 把 DIB 文件转换为 DDB 的方法。

程序 15-6　DIBCONV 程序

DIBCONV.C

```
/*--------------------
    DIBCONV.C——Converts a DIB to a DDB
                (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>
#include <commdlg.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("DibConv") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
```

```
            MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                        szAppName, MB_ICONERROR) ;
            return 0 ;
        }

        hwnd = CreateWindow (szAppName, TEXT ("DIB to DDB Conversion"),
                             WS_OVERLAPPEDWINDOW,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                             CW_USEDEFAULT, CW_USEDEFAULT,
                             NULL, NULL, hInstance, NULL) ;

        ShowWindow (hwnd, iCmdShow) ;
        UpdateWindow (hwnd) ;

        while (GetMessage (&msg, NULL, 0, 0))
        {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;

        }
        return msg.wParam ;
}

HBITMAP CreateBitmapObjectFromDibFile (HDC hdc, PTSTR szFileName)
{
        BITMAPFILEHEADER * pbmfh ;
        BOOL               bSuccess ;
        DWORD              dwFileSize, dwHighSize, dwBytesRead ;
        HANDLE             hFile ;
        HBITMAP            hBitmap ;

            // Open the file; read access, prohibit write access

        hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                            OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
        if (hFile == INVALID_HANDLE_VALUE)
            return NULL ;

            // Read in the whole file

        dwFileSize = GetFileSize (hFile, &dwHighSize) ;

        if (dwHighSize)
        {
            CloseHandle (hFile) ;
            return NULL ;
        }
```

```
    pbmfh = malloc (dwFileSize) ;

    if (!pbmfh)
    {
         CloseHandle (hFile) ;
         return NULL ;
    }

    bSuccess = ReadFile (hFile, pbmfh, dwFileSize, &dwBytesRead, NULL) ;
    CloseHandle (hFile) ;

        // Verify the file

    if (!bSuccess || (dwBytesRead != dwFileSize)
                  || (pbmfh -> bfType != * (WORD * ) "BM")
                  || (pbmfh -> bfSize != dwFileSize))
    {
         free (pbmfh) ;
         return NULL ;
    }
        // Create the DDB

    hBitmap = CreateDIBitmap (hdc,
                              (BITMAPINFOHEADER * ) (pbmfh + 1),
                              CBM_INIT,
                              (BYTE * ) pbmfh + pbmfh -> bfOffBits,
                              (BITMAPINFO * ) (pbmfh + 1),
                              DIB_RGB_COLORS) ;
    free (pbmfh) ;

    return hBitmap ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HBITMAP          hBitmap ;
    static int              cxClient, cyClient ;
    static OPENFILENAME     ofn ;
    static TCHAR            szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
    static TCHAR            szFilter[ ] = TEXT ("Bitmap Files ( * .BMP) \ 0 * .bmp \ 0")
                                          TEXT ("All Files ( * . * ) \ 0 * . * \ 0 \ 0") ;
    BITMAP                  bitmap ;
    HDC                     hdc, hdcMem ;
    PAINTSTRUCT             ps ;

    switch (message)
    {
    case WM_CREATE:
```

```
        ofn.lStructSize          = sizeof (OPENFILENAME) ;
        ofn.hwndOwner            = hwnd ;
        ofn.hInstance            = NULL ;
        ofn.lpstrFilter          = szFilter ;
        ofn.lpstrCustomFilter    = NULL ;
        ofn.nMaxCustFilter       = 0 ;
        ofn.nFilterIndex         = 0 ;
        ofn.lpstrFile            = szFileName ;
        ofn.nMaxFile             = MAX_PATH ;
        ofn.lpstrFileTitle       = szTitleName ;
        ofn.nMaxFileTitle        = MAX_PATH ;
        ofn.lpstrInitialDir      = NULL ;
        ofn.lpstrTitle           = NULL ;
        ofn.Flags                = 0 ;
        ofn.nFileOffset          = 0 ;
        ofn.nFileExtension       = 0 ;
        ofn.lpstrDefExt          = TEXT ("bmp") ;
        ofn.lCustData            = 0 ;
        ofn.lpfnHook             = NULL ;
        ofn.lpTemplateName       = NULL ;

        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing DIB, delete it

            if (hBitmap)
            {
                DeleteObject (hBitmap) ;
                hBitmap = NULL ;
            }
                // Create the DDB from the DIB

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
```

```
            ShowCursor (TRUE) ;

            hdc = GetDC (hwnd) ;
            hBitmap = CreateBitmapObjectFromDibFile (hdc, szFileName) ;
            ReleaseDC (hwnd, hdc) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                // Invalidate the client area for later update

            InvalidateRect (hwnd, NULL, TRUE) ;
            if (hBitmap == NULL)
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
            }
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hBitmap)
        {
            GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;

            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;

            BitBlt (hdc,      0, 0, bitmap.bmWidth, bitmap.bmHeight,
                    hdcMem, 0, 0, SRCCOPY) ;

            DeleteDC (hdcMem) ;
        }

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (hBitmap)
            DeleteObject (hBitmap) ;
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIBCONV.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

DIBCONV MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",               IDM_FILE_OPEN
    END
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by DibConv.rc

# define IDM_FILE_OPEN              40001
```

DIBCONV.C是自包含的并且不需要前面的文件。在对它仅有的菜单命令（File Open）的响应中，WndProc 调用程序的 CreateBitmapObjectFromDibFile 函数。这个函数将整个文件读入内存，并将指向内存块的指针传递给 CreateDIBitmap 函数。该函数返回到位图的句柄，然后包含 DIB 的内存块即被释放。在处理 WM_PAINT 消息期间，WndProc 将位图选入兼容的内存设备描述表，并使用 BitBlt（不是 SetDIBitsToDevice）在客户区显示位图。它通过使用位图句柄调用带有 BITMAP 结构的 GetObject 函数来获取位图的宽度和高度。

在从 CreateDIBitmap 创建位图时不必初始化 DDB 像素位，之后可以调用 SetDIBits 初始化像素位。该函数的语法如下：

```
iLines = SetDIBits (
            hdc,            // device context handle
            hBitmap,        // bitmap handle
            yScan,          // first scan line to convert
            cyScans,        // number of scan lines to convert
            pBits,          // pointer to pixel bits
            pInfo,          // pointer to DIB information
            fClrUse) ;      // color use flag
```

此函数使用了 BITMAPINFO 结构中的色彩表把这些位转换为设备相关的格式。仅当最后一个参数设置为 DIB_PAL_COLORS 时，才需要设备描述表句柄。

15.3.2 从 DDB 到 DIB

与 SetDIBits 函数相似的函数是 GetDIBits,您可以使用此函数把 DDB 转化为 DIB:

```
int WINAPI GetDIBits (
          hdc,          // device context handle
          hBitmap,      // bitmap handle
          yScan,        // first scan line to convert
          cyScans,      // number of scan lines to convert
          pBits,        // pointer to pixel bits (out)
          pInfo,        // pointer to DIB information (out)
          fClrUse) ;    // color use flag
```

然而我认为,仅仅简单地把此函数看成是 SetDIBits 的逆运算那是不够的。一般情况下,如果使用 CreateDIBitmap 和 SetDIBits 将 DIB 转换为 DDB,然后使用 GetDIBits 把 DDB 转换回 DIB,您就不会得到原来的图像。这是因为在 DIB 被转换为设备相关的格式时,有一些信息丢失了。丢失信息的数量取决于进行转换时 Windows 所运行的视频模式。

您可能会发现没有使用 GetDIBits 的必要性。考虑一下:在什么环境下您的程序发现自身带有位图句柄,但没有用于在起始的位置创建位图的数据?那么剪贴板如何呢?但是剪贴板为 DIB 提供了自动的转换。您所做的屏幕捕获,例如第十四章中 BLOWUP 程序所做的,就是 GetDIBits 函数的一个例子,我不演示这个函数,但在"知识库(Knowledge Base)"文章 Q80080 中有一些信息。

15.3.3 DIB 块

希望您已经对设备相关和设备无关位图的区别有了清晰的概念。DIB 能拥有几种色彩组织中的一种,DDB 必须是单色的或是与真实输出设备相同的格式。DIB 是一个文件或内存块;DDB 是 GDI 位图对象并由位图句柄表示。DIB 能被显示或转换为 DDB,并转换回 DIB,但是这里包含了设备无关位和设备相关位之间的转换过程。

现在将遇到一个函数,它打破了这些规则。该函数在 32 位 Windows 版本中引入,称为 CreateDIBSection,语法为

```
hBitmap = CreateDIBSection (
          hdc,          // device context handle
          pInfo,        // pointer to DIB information
          fClrUse,      // color use flag
          ppBits,       // pointer to pointer variable
          hSection,     // file-mapping object handle
          dwOffset) ;   // offset to bits in file-mapping object
```

CreateDIBSection 是 Windows API 中最重要的函数之一(至少在使用位图时),然而您会发

现它很深奥，并难以理解。

让我们从它的名称开始，我们知道 DIB 是什么，但“DIB section”到底是什么呢？当您第一次检查 CreateDIBSection 时，可能会寻找该函数与 DIB 部分工作的方式。这是正确的，CreateDIBSection 所做的就是创建 DIB 的一部分（位图像素位的内存块）。

现在我们看一下返回值，它是 GDI 位图对象的句柄，这个返回值可能是该函数调用最具欺骗性的部分。返回值似乎暗示着 CreateDIBSection 在功能上与 CreateDIBitmap 相同，事实上，它只是相似，但完全不同。实际上，从 CreateDIBSection 返回的位图句柄与我们在本章和上一章遇到的所有位图创建函数返回的位图句柄在本质上不同。

一旦理解了 CreateDIBSection 的真实特性，您可能奇怪为什么不把返回值定义得有所区别。您也可能得出结论：CreateDIBSection 应该被称为“CreateDIBitmap”，并且如同前面所指出的，CreateDIBitmap 应该被称为“CreateDDBitmap”。

先让我们考察一下如何简化 CreateDIBSection，并正确地使用它。首先，把最后两个参数 hSection 和 dwOffset，分别设置为 NULL 和 0，我将在本章最后讨论这些参数的用法。第二，仅在 fColorUse 参数设置为 DIB_PAL_COLORS 时，才使用 hdc 参数，如果 fColorUse 为 DIB_RGB_COLORS（或 0），hdc 将被忽略。（这与 CreateDIBitmap 不同，hdc 参数用于获取与 DDB 兼容的设备的色彩格式。）

因此，CreateDIBSection 的最简单的形式仅需要第 2 和第 4 个参数。第 2 个参数是指向 BITMAPINFO 结构的指针，我们以前曾使用过。我希望第 4 个参数的指向指针定义的指针不会使您困惑，实际上它很简单。

假定要创建每像素 24 位的 384 × 256 位 DIB，24 位格式不需要色彩表，因此它是最简单的，因此可以为 BITMAPINFO 参数使用 BITMAPINFOHEADER 结构。

需要定义三个变量：BITMAPINFOHEADER 结构、BYIE 指针和位图句柄：

```
BITMAPINFOHEADER bmih ;
BYTE             * pBits ;
HBITMAP            hBitmap ;
```

现在初始化 BITMAPINFOHEADER 结构的字段：

```
bmih -> biSize           = sizeof (BITMAPINFOHEADER) ;
bmih -> biWidth          = 384 ;
bmih -> biHeight         = 256 ;
bmih -> biPlanes         = 1 ;
bmih -> biBitCount       = 24 ;
bmih -> biCompression    = BI_RGB ;
bmih -> biSizeImage      = 0 ;
bmih -> biXPelsPerMeter  = 0 ;
bmih -> biYPelsPerMeter  = 0 ;
bmih -> biClrUsed        = 0 ;
bmih -> biClrImportant   = 0 ;
```

经过最小量的准备，我们调用该函数：

```
hBitmap = CreateDIBSection (NULL, (BITMAPINFO *) &bmih, 0, &pBits, NULL, 0) ;
```

注意，我们为第 2 个参数赋予 BITMAPINFOHEADER 结构的地址，这是很平常的，但 BYTE 指针 pBits 的地址，就不平常了。这样，第 4 个参数是函数需要的指向指针的指针。

这是函数调用所做的：CreateDIBSection 检查 BITMAPINFOHEADER 结构，并分配足够的内存块来加载 DIB 像素位。（在这个例子里，块的大小为 384 × 256 × 3 字节。）它在您提供的 pBits 参数中存储了指向此内存块的指针。函数返回位图句柄，正如我所说的，它与 CreateDIBitmap 和其他位图创建函数返回的句柄不一样。

然而，我们还没有做完，位图像素是未初始化的。如果正在读取 DIB 文件，可以简单地把 pBits 参数传递给 ReadFile 函数并读取它们，也可以使用一些程序代码“手工”设置。

程序 15-7 所示的 DIBSECT 除了调用 CreateDIBSection 而不是 CreateDIBitmap 之外，与 DIBCONV 程序相似。

程序 15-7　DIBSECT 程序

DIBSECT.C

```
/*-------------------------
   DIBSECT.C——Displays a DIB Section in the client area
              (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include <commdlg.h>
#include "resource.h"
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("DibSect") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
```

```
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("DIB Section Display"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;

     }
     return msg.wParam ;
}

HBITMAP CreateDibSectionFromDibFile (PTSTR szFileName)
{
     BITMAPFILEHEADER  bmfh ;
     BITMAPINFO      * pbmi ;
     BYTE            * pBits ;
     BOOL              bSuccess ;
     DWORD             dwInfoSize, dwBytesRead ;
     HANDLE            hFile ;
     HBITMAP           hBitmap ;

          // Open the file: read access, prohibit write access

     hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ,
                         NULL, OPEN_EXISTING, 0, NULL) ;

     if (hFile == INVALID_HANDLE_VALUE)
```

```
    return NULL ;

// Read in the BITMAPFILEHEADER

bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                     &dwBytesRead, NULL) ;

if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEADER))
              || (bmfh.bfType != * (WORD *) "BM"))
{
     CloseHandle (hFile) ;
     return NULL ;
}

    // Allocate memory for the BITMAPINFO structure & read it in

dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER) ;

pbmi = malloc (dwInfoSize) ;

bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesRead, NULL) ;

if (!bSuccess || (dwBytesRead != dwInfoSize))
{
     free (pbmi) ;
     CloseHandle (hFile) ;
     return NULL ;
}
    // Create the DIB Section

hBitmap = CreateDIBSection (NULL, pbmi, DIB_RGB_COLORS, &pBits, NULL, 0) ;

if (hBitmap == NULL)
{
     free (pbmi) ;
     CloseHandle (hFile) ;
     return NULL ;
}

    // Read in the bitmap bits

ReadFile (hFile, pBits, bmfh.bfSize - bmfh.bfOffBits, &dwBytesRead, NULL) ;

free (pbmi) ;
CloseHandle (hFile) ;

return hBitmap ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP          hBitmap ;
     static int              cxClient, cyClient ;
     static OPENFILENAME     ofn ;
     static TCHAR            szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR            szFilter[] = TEXT ("Bitmap Files (*.BMP) \0*.bmp \0")
                                          TEXT ("All Files (*.*) \0*.* \0\0") ;
     BITMAP                  bitmap ;
     HDC                     hdc, hdcMem ;
     PAINTSTRUCT             ps ;

     switch (message)
     {
     case WM_CREATE:
          ofn.lStructSize       = sizeof (OPENFILENAME) ;
          ofn.hwndOwner         = hwnd ;
          ofn.hInstance         = NULL ;
          ofn.lpstrFilter       = szFilter ;
          ofn.lpstrCustomFilter = NULL ;
          ofn.nMaxCustFilter    = 0 ;
          ofn.nFilterIndex      = 0 ;
          ofn.lpstrFile         = szFileName ;
          ofn.nMaxFile          = MAX_PATH ;
          ofn.lpstrFileTitle    = szTitleName ;
          ofn.nMaxFileTitle     = MAX_PATH ;
          ofn.lpstrInitialDir   = NULL ;
          ofn.lpstrTitle        = NULL ;
          ofn.Flags             = 0 ;
          ofn.nFileOffset       = 0 ;
          ofn.nFileExtension    = 0 ;
          ofn.lpstrDefExt       = TEXT ("bmp") ;
          ofn.lCustData         = 0 ;
          ofn.lpfnHook          = NULL ;
          ofn.lpTemplateName    = NULL ;

          return 0 ;

     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_COMMAND:
          switch (LOWORD (wParam))
          {
          case IDM_FILE_OPEN:
```

```
                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing bitmap, delete it

            if (hBitmap)
            {
                DeleteObject (hBitmap) ;
                hBitmap = NULL ;
            }
                // Create the DIB Section from the DIB file

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            hBitmap = CreateDIBsectionFromDibFile (szFileName) ;
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                // Invalidate the client area for later update

            InvalidateRect (hwnd, NULL, TRUE) ;

            if (hBitmap == NULL)
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
            }
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hBitmap)
        {
            GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;

            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;

            BitBlt (hdc,      0, 0, bitmap.bmWidth, bitmap.bmHeight,
                    hdcMem, 0, 0, SRCCOPY) ;

            DeleteDC (hdcMem) ;
```

```
        }

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (hBitmap)
            DeleteObject (hBitmap) ;

        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIBSECT.RC(摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

DIBSECT MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",               IDM_FILE_OPEN
    END
END
```

RESOURCE.H(摘录)

```
// Microsoft Developer Studio generated include file.
// Used by DIBsect.rc

# define IDM_FILE_OPEN              40001
```

注意 DIBCONV 中 CreateBitmapObjectFromDibFile 函数和 DIBSECT 中 CreateDIbsectionFromDibFile 函数之间的区别。DIBCONV 读入整个文件，然后把指向 DIB 内存块的指针传递给 CreateDIBitmap 函数。DIBSECT 首先读取 BITMAPFILEHEADER 结构中的信息，然后确定 BITMAPINFO 结构的大小，为此分配内存，并在第二个 ReadFile 调用中将它读入内存。然后，函数把指向 BITMAPINFO 结构和指针变量 pBits 的指针传递给 CreateDIBSection。函数返回位图句柄并设置 pBits 指向函数将要读取 DIB 像素位的内存块。

pBits 指向的内存块归系统所有。当通过调用 DeleteObject 删除位图时内存被自动

释放。然而，程序能利用该指针直接改变 DIB 位。当应用程序通过 API 传递大量内存块时，系统拥有这些内存块在 WINDOWS NT 下不会影响速度。

我上面曾说过，当在视频显示器上显示 DIB 时，某些时候必须进行从设备无关像素到设备相关像素的转换，有时这些格式转换可能非常长。我们看一看三种用于显示 DIB 的方法：

- 当使用 SetDIBitsToDevice 或 StretchDIBits 来把 DIB 直接显示在屏幕上，格式转化在 SetDIBitsToDevice 或 StretchDIBits 调用期间发生；
- 当使用 CreateDIBitmap 和（可能是）SetDIBits 把 DIB 转化为 DDB，然后使用 BitBlt 或 StretchBlt 来显示它时，如果设置了 CBM_INIT 标志，则格式转化在 CreateDIBitmap 或 SetDIBits 期间发生；
- 当使用 CreateDIBSection 创建 DIB 部分，然后使用 BitBlt 或 StretchBlt 显示它时，则格式转化在 BitBlt 对 StretchBlt 的调用期间发生。

再读一下最后一个句子，从 CreateDIBSection 返回的位图句柄不同于我们所遇到的其他位图句柄。此位图句柄实际上指向存储在内存中由系统维护但应用程序能访问的 DIB。在需要的时候 DIB 会转化为特定的色彩格式，通常是在用 BitBlt 或 StretchBlt 显示位图时。

您也可以将位图句柄选入内存设备描述表，并使用 GDI 函数来绘制。在 pBits 变量指向的 DIB 像素内将反映出结果。因为 Windows NT 下的 GDI 函数分批调用，在内存设备背景上绘制之后和“手工”的访问位之前调用 GdiFlush 。

在 DIBSECT，我们清除 pBits 变量，因为程序不再需要变量。如果需要直接更改位，那么这是使用 CreateDIBSection 的主要原因。在 CreateDIBSection 调用之后就没有方法来获取位指针。

15.3.4　DIB 块的其他区别

从 CreateDIBitmap 返回的位图句柄与函数的 hdc 参数引用的设备有相同的平面和像素位组织。您能通过具有 BITMAP 结构的 GetObject 调用来检验这一点。

CreateDIBSection 就不同了，如果在从该函数返回的位图句柄的基础上带有 BITMAP 结构调用 GetObject，您会发现位图具有的色彩组织与 BITMAPINFOHEADER 结构的字段指出的色彩组织相同。但是您能将这个句柄选入与视频显示器兼容的内存设备描述表。这与上一章关于 DDB 的内容相矛盾，但这也就是我说此 DIB 部分位图句柄不同的原因。

另一个奇妙之处是：您可能还记得，DIB 中像素数据行的字节长度始终是 4 的倍数。GDI 位图对象中行的字节长度，就是使用 GetObject 从 BITMAP 结构的 bmWidthBytes 字段中得到的长度，始终是 2 的倍数。如果按每像素 24 位而宽度为 2 像素设置 BITM-

APINFOHEADER 结构并随后调用 GetObject，您就会发现 bmWidthBytes 字段是 8 而不是 6。

使用从 CreateDIBSection 返回的位图句柄，也可以带有 DIBSECTION 结构调用 GetObject：

```
GetObject (hBitmap, sizeof (DIBSECTION), &dibsection) ;
```

如果位图句柄是从其他位图创建函数返回的，则此函数也不会运行。DIBSECTION 结构定义如下：

```
typedef struct tagDIBSECTION     // ds
{
     BITMAP              dsBm ;              // BITMAP structure
     BITMAPINFOHEADER    dsBmih ;            // DIB information header
     DWORD               dsBitfields [3] ;   // color masks
     HANDLE              dshSection ;        // file-mapping object handle
     DWORD               dsOffset ;          // offset to bitmap bits
}
DIBSECTION, * PDIBSECTION ;
```

此结构包含 BITMAP 结构和 BITMAPINFOHEADER 结构。最后两个字段是传递给 CreateDIBSection 的最后两个参数，一会儿我会介绍它们。

DIBSECTION 结构中包含了除了色彩表以外的有关位图的许多内容。当把 DIB 部分位图句柄选入内存设备描述表时，可以通过调用 GetDIBColorTable 来得到色彩表：

```
hdcMem = CreateCompatibleDC (NULL) ;
SelectObject (hdcMem, hBitmap) ;
GetDIBColorTable (hdcMem, uFirstIndex, uNumEntries, &rgb) ;
DeleteDC (hdcMem) ;
```

同样，您可以通过调用 SetDIBColorTable 来设置色彩表中的条目。

15.3.5　文件映射选项

我们还没有讨论 CreateDIBSection 的最后两个参数，它们是文件映射对象的句柄和文件中位图位开始的偏移量。文件映射对象使您能够像文件位于内存中一样处理文件。也就是说，可以通过使用内存指针来访问文件，但文件不需要整个位于内存。

在大 DIB 的情况下，此技术有助于减少内存需求。DIB 像素位能够存储在磁盘上，但仍然可以当作位于内存中一样进行访问，虽然会影响性能。问题是，当像素位实际上存储在磁盘上时，它们不可能是实际 DIB 文件的一部分。它们必须位于其他的文件内。

为了演示这个过程，下面显示的函数除了不把像素位读入内存以外，其他与 DIBSECT 中创建的 DIB 部分的函数很相似，然而它提供了文件映射对象和传递给 CreateDIBSection 函数的偏移量：

```
HBITMAP CreateDIBsectionMappingFromFile (PTSTR szFileName)
{
     BITMAPFILEHEADER  bmfh ;
     BITMAPINFO      * pbmi ;
     BYTE            * pBits ;
     BOOL              bSuccess ;
     DWORD             dwInfoSize, dwBytesRead ;
     HANDLE            hFile, hFileMap ;
     HBITMAP           hBitmap ;

     hFile = CreateFile (szFileName, GENERIC_READ | GENERIC_WRITE,
                         0,                  // No sharing!
                         NULL, OPEN_EXISTING, 0, NULL) ;

     if (hFile == INVALID_HANDLE_VALUE)
          return NULL ;

     bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                          &dwBytesRead, NULL) ;

     if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEADER))
                   || (bmfh.bfType != * (WORD *) "BM"))
     {
          CloseHandle (hFile) ;
          return NULL ;
     }
     dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER) ;
     pbmi = malloc (dwInfoSize) ;
     bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesRead, NULL) ;

     if (!bSuccess || (dwBytesRead != dwInfoSize))
     {
          free (pbmi) ;
          CloseHandle (hFile) ;
          return NULL ;
     }
     hFileMap = CreateFileMapping (hFile, NULL, PAGE_READWRITE, 0, 0, NULL) ;

     hBitmap = CreateDIBSection (NULL, pbmi, DIB_RGB_COLORS, &pBits,
                                 hFileMap, bmfh.bfOffBits) ;
     free (pbmi) ;
     return hBitmap ;
}
```

啊哈，它不能工作。CreateDIBSection 的文档指出“dwOffset(函数的最后一个参数)必须是 DWORD 大小的倍数”。尽管信息头的大小始终是 4 的倍数，并且色彩表的大小也始终是 4 的倍数，但位图文件头却不是，它是 14 字节。因此 bmfh.bfOffBits 永远不会是

4 的倍数。

15.3.6 总结

如果有小的 DIB,并且需要频繁地操作像素位,您可以使用 SetDIBitsToDevice 和 StretchDIBits 来显示它们。然而,对于大的 DIB,此技术会遇到性能问题,尤其在 8 位视频显示器上和 Windows NT 环境下。

您可以使用 CreateDIBitmap 和 SetDIBits 把 DIB 转化为 DDB。现在,显示位图可以使用快速的 BitBlt 和 StretchBlt 函数。然而,您不能再访问与设备无关的像素位。

CreateDIBSection 是一个很好的折中方案。在 Windows NT 下通过 BitBlt 和 StretchBlt 使用位图句柄比使用 SetDIBitsToDevice 和 StretchDIBits(但没有 DDB 的缺陷)会得到更好的性能。您仍然可以访问 DIB 像素位。

在下一章,我们将在讨论"Windows 调色板管理器"之后完成位图的探索。

第十六章

调色板管理器

如果硬件允许,本章就没有存在的必要。尽管许多现代的视频适配器板提供 24 位颜色(也称“真彩色”或“数百万色”)或 16 位颜色(“增强色”或“数千种色”),一些视频适配器——尤其是在便携式电脑上或高分辨率模式中——每个像素只允许 8 位。这意味着仅有 256 种颜色。

我们用 256 种颜色能做什么呢? 很明显,要显示真实世界的图像,仅 16 种颜色显然不够,用数千或数百万种颜色则足以把色彩斑斓的现实世界描绘得淋漓尽致,256 种颜色位于中间状态。是的,用 256 种颜色来显示真实世界的图像足够了,但需要根据特定的图像来指定这些颜色。这意味着操作系统不能简单地选择“标准”系列的 256 种颜色,就希望它们对每个应用程序都是理想的颜色。

这就是 Windows 调色板管理器所要涉及的全部内容。它用于指定程序在 8 位视频模式下运行时所需要的颜色。如果知道程序肯定不会在 8 位视频模式下运行,那么您也不需要使用调色板管理器。不过,由于补充了位图的一些细节,所以本章还是包含重要信息的。

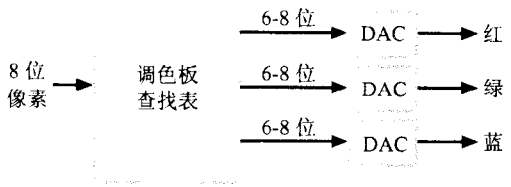
16.1 使用调色板

传统上讲,调色板是画家用来混合颜色的板。这个词也可以指画家在绘画过程中使用的所有颜色。在计算机图形中,调色板是在图形输出设备(例如视频显示器)上可用的颜色范围。该词也可以指支持 256 色模式的显示卡上的查找表。

16.1.1 视频硬件

显示卡上的调色板查找表工作过程如下页图所示。在 8 位视频模式中,每个像素占 8 位。像素值指定了查找表(包含 256 个 RGB 值)中的地址。这些 RGB 值可以正好 24 位

宽，或者小一点，通常是 18 位宽（即主要的红、绿和蓝各 6 位）。每种颜色的值都输入到数字模拟转换器，以得到发送给监视器的红、绿和蓝三个模拟信号。



通常，软件可以用任意值来加载调色板查找表，但这对设备无关的窗口界面，例如 Microsoft Windows，会有一些干扰。首先，Windows 必须提供软件接口，以便在不直接涉及硬件的情况下，应用程序就可以访问调色板管理器。第二个问题更严重：因为所有的应用程序都共享同一个视频显示器，而且同时运行，所以一个应用程序使用了调色板查找表可能会影响其他程序的使用。

这时就需要使用 Windows 调色板管理器（在 Windows 3.0 中提出）。Windows 保留了 256 种颜色中的 20 种，而允许应用程序修改其余的 236 种。（在某些情况下，应用程序最多可以改变 256 种颜色中的 254 种——只有黑色和白色除外——但这有一点麻烦）。Windows 为系统保留的 20 种颜色（有时称为 20 种“静态”颜色）如表 16-1 所示。

表 16-1　256 色视频模式中的 20 种保留颜色

像素位	RGB 值	颜色名称	像素位	RGB 值	颜色名称
00000000	00 00 00	黑	11111111	FF FF FF	白
00000001	80 00 00	深红	11111110	00 FF FF	青
00000010	00 80 00	深绿	11111101	FF 00 FF	品红
00000011	80 80 00	深黄	11111100	00 00 FF	蓝
00000100	00 00 80	深蓝	11111011	FF FF 00	黄
00000101	80 00 80	深品红	11111010	00 FF 00	绿
00000110	00 80 80	深青	11111001	FF 00 00	红
00000111	C0 C0 C0	浅灰	11111000	80 80 80	深灰
00001000	C0 DC C0	美元绿	11110111	A0 A0 A4	中性灰
00001001	A6 CA F0	天蓝	11110110	FF FB F0	乳白色

在 256 色视频模式下运行时，由 Windows 维护系统调色板，此调色板与显示卡上的硬件调色板查找表相同。默认的系统调色板如表 16-1 所示。应用程序可以通过指定“逻辑调色板（logical palettes）”来修改其余 236 种颜色。如果有多个应用程序使用逻辑

调色板，那么 Windows 就给活动窗口最高级优先权。（我们知道，活动窗口有高亮显示标题栏，并且显示在其他所有窗口的前面。）我们将用一些简单的示例程序来检查它是如何工作的。

要运行本章其他部分的程序，您可能需要将显示卡切换成 256 色模式。在桌面上右键单击鼠标，从菜单中选择“属性”，然后选择“设置”选项卡。

16.1.2　显示渐变灰色

程序 16-1 所示的 GRAYS1 程序没有使用 Windows 调色板管理器，而尝试用正常显示的 65 级渐变灰色作为从黑到白的多种色彩的“来源”。

程序 16-1　GRAYS1 程序

GRAYS1.C

```
/*-------------------
   GRAYS1.C——Gray Shades
            (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Grays1") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
```

```
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Shades of Gray # 1"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static int      cxClient, cyClient ;
    HBRUSH          hBrush ;
    HDC             hdc ;
    int             i ;
    PAINTSTRUCT     ps ;
    RECT            rect ;

    switch (message)
    {
    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

            // Draw the fountain of grays

        for (i = 0 ; i < 65 ; i++)
        {
            rect.left   = i * cxClient / 65 ;
            rect.top    = 0 ;
            rect.right  = (i + 1) * cxClient / 65 ;
```

```
            rect.bottom = cyClient ;

            hBrush = CreateSolidBrush (RGB (min (255, 4 * i),
                                            min (255, 4 * i),
                                            min (255, 4 * i))) ;
            FillRect (hdc, &rect, hBrush) ;
            DeleteObject (hBrush) ;

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在 WM_PAINT 消息期间，程序调用了 65 次 FillRect 函数，每次都使用不同级别的灰色创建的画刷。灰色值是 RGB 值(0,0,0)、(4,4,4)、(8,8,8)，等等，直到最后一个值(255,255,255)。最后一个值来自 CreateSolidBrush 函数中的 min 宏。

如果在 256 色视频模式下运行该程序，您将看到从黑到白的 65 种灰色条纹，而且它们几乎都用混合着色。纯颜色只有黑色、深灰色(128,128,128)、浅灰色(192,192,192)和白色。其他颜色是混合了这些纯颜色的多位模式。如果我们在显示行或文本，而不是用这 65 种灰色底纹填充区域，Windows 将不使用混合色而只使用这 4 种纯色。如果我们正在显示位图，则图像将用 20 种标准 Windows 颜色近似。这时正如您在运行最后一章中的程序的同时又加载了彩色或灰色底纹 DIB 所见到的一样。通常，Windows 在位图中不使用混合色。

程序 16-2 所示的 GRAYS2 程序用较少的外部代码验证了调色板管理器中最重要的函数和消息。

程序 16-2　GRAYS2 程序

GRAYS2.C

```
/*-----------------------
   GRAYS2.C——Gray Shades Using Palette Manager
             (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Grays2") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Shades of Gray #2"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HPALETTE  hPalette ;
     static int       cxClient, cyClient ;
     HBRUSH           hBrush ;
```

```
HDC             hdc ;
int             i ;
LOGPALETTE  *   plp ;
PAINTSTRUCT     ps ;
RECT            rect ;

switch (message)
{
case WM_CREATE:
        // Set up a LOGPALETTE structure and create a palette

    plp = malloc (sizeof (LOGPALETTE) + 64 * sizeof (PALETTEENTRY)) ;

    plp -> palVersion    = 0x0300 ;
    plp -> palNumEntries = 65 ;

    for (i = 0 ; i < 65 ; i++)
    {
        plp -> palPalEntry[i].peRed   = (BYTE) min (255, 4 * i) ;
        plp -> palPalEntry[i].peGreen = (BYTE) min (255, 4 * i) ;
        plp -> palPalEntry[i].peBlue  = (BYTE) min (255, 4 * i) ;
        plp -> palPalEntry[i].peFlags = 0 ;
    }
    hPalette = CreatePalette (plp) ;
    free (plp) ;
    return 0 ;

case WM_SIZE:
    cxClient = LOWORD (lParam) ;
    cyClient = HIWORD (lParam) ;
    return 0 ;

case WM_PAINT:
    hdc = BeginPaint (hwnd, &ps) ;

        // Select and realize the palette in the device context

    SelectPalette (hdc, hPalette, FALSE) ;
    RealizePalette (hdc) ;

        // Draw the fountain of grays

    for (i = 0 ; i < 65 ; i++)
    {
        rect.left   = i * cxClient / 64 ;
        rect.top    = 0 ;
        rect.right  = (i + 1) * cxClient / 64 ;
        rect.bottom = cyClient ;
```

```
                hBrush = CreateSolidBrush (PALETTERGB (min (255, 4 * i),
                                                       min (255, 4 * i),
                                                       min (255, 4 * i))) ;
                FillRect (hdc, &rect, hBrush) ;
                DeleteObject (hBrush) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_QUERYNEWPALETTE:
            if (!hPalette)
                return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

        case WM_PALETTECHANGED:
            if (!hPalette || (HWND) wParam == hwnd)
                break ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            UpdateColors (hdc) ;

            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
            DeleteObject (hPalette) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

通常，使用调色板管理器的第一步就是调用 CreatePalette 函数来创建逻辑调色板。逻辑调色板包含程序所需要的全部颜色——即 236 种颜色。GRAYS1 程序在 WM_CREATE 消息期间处理此作业。它初始化 LOGPALETTE("logical palette(逻辑调色板)")结构的字段，并将这个结构的指针传递给 CreatePalette 函数。CreatePalette 返回逻辑调色板的句柄，并将此句柄保存在静态变量 hPalette 中。

LOGPALETTE 结构定义如下：

```
typedef struct
{
    WORD        palVersion ;
    WORD        palNumEntries ;
    PALETTEENTRY    palPalEntry[1] ;
}
LOGPALETTE, * PLOGPALETTE ;
```

第一个字段通常设为 0x0300，表示兼容 Windows 3.0；第二个字段设置为调色板表中的条目数；LOGPALETTE 结构中的第三个字段是一个 PALETTEENTRY 结构的数组，此结构也是一个调色板条目。PALETTEENTRY 结构定义如下：

```
typedef struct
{
    BYTE peRed ;
    BYTE peGreen ;
    BYTE peBlue ;
    BYTE peFlags ;
}
PALETTEENTRY, * PPALETTEENTRY ;
```

每个 PALETTEENTRY 结构都定义了一个我们要在调色板中使用的 RGB 颜色值。

注意，LOGPALETTE 中只能定义一个 PALETTEENTRY 结构的数组。您需要为 LOGPALETTE 结构和附加的 PALETTEENTRY 结构分配足够大的内存空间。GRAYS2 需要 65 种灰色底纹，因此它为 LOGPALETTE 结构和 64 个附加的 PALETTEENTRY 结构分配了足够大的内存空间。GRAYS2 将 palNumEntries 字段设置为 65，然后从 0 到 64 循环，计算灰度等级（一般是循环索引的 4 倍，但不超过 255），将结构中的 peRed、peGreen 和 peBlue 字段设置为此灰度等级。peFlags 字段设为 0。程序将指向这个内存块的指针传递给 CreatePalette，在一个静态变量中保存该调色板句柄，然后释放内存。

逻辑调色板是 GDI 对象。程序应该删除它们创建的全部逻辑调色板。WndProc 通过在 WM_DESTROY 消息期间调用 DeleteObject，仔细地删除逻辑调色板。

注意逻辑调色板是独立的设备描述表。在真正使用之前，必须确保将其选进设备描述表。在 WM_PAINT 消息期间，SelectPalette 将逻辑调色板选进设备描述表。除了含有第三个参数以外，此函数与 SelectObject 函数相似。通常，第三个参数设为 FALSE。如果 SelectPalette 的第三个参数设为 TRUE，那么调色板将始终是“后台调色板”，这意味着当其他所有程序都实现了各自的调色板之后，该调色板才可以获得仍位于系统调色板中的一个未使用条目。

在任何时候都只能有一个逻辑调色板选进设备描述表。函数将返回前一个选进设备描述表的逻辑调色板句柄。如果您希望将此逻辑调色板重新选进设备描述表，则可

以保存此句柄。

通过将颜色映射到系统调色板，RealizePalette 函数使 Windows 在设备描述表中“实现”逻辑调色板，而系统调色板是与显示卡实际的物理调色板相对应。实际工作在此函数调用期间进行。Windows 必须决定调用函数的窗口是活动的还是非活动的，并尽可能通知其他窗口系统调色板已改变。（我们将简要说明一下通知的过程。）

回忆一下 GRAYS1，它用 RGB 宏来指定纯色画刷的颜色。RGB 宏构造一个 32 位长整数（记作 COLORREF 值），其中高位字节是 0，3 个低位字节是红、绿和蓝的亮度。

使用 Windows 调色板管理器的程序可以继续使用 RGB 颜色值来指定颜色。不过，这些 RGB 颜色值将不能访问逻辑调色板中的附加颜色。它们的作用与没有使用调色板管理器相同。要在逻辑调色板中使用附加的颜色，就要用到 PALETTERGB 宏。除了 COLORREF 值的高位字节设为 2 而不是 0 以外，“调色板 RGB”颜色与 RGB 颜色很相似。

下面是重要的规则：

- 为了使用逻辑调色板中的颜色，请用调色板 RGB 值或调色板索引来指定。（我将简要介绍调色板索引。）不要使用常规的 RGB 值。如果使用了常规的 RGB 值，将得到一种标准颜色，而不是逻辑调色板中的颜色。
- 没有将调色板选进设备描述表时，不要使用调色板 RGB 值或调色板索引。
- 尽管可以使用调色板 RGB 值来指定逻辑调色板中没有的颜色，但您还是从逻辑调色板获得颜色。

例如，在 GRAYS2 中处理 WM_PAINT 期间，选择并实现了逻辑调色板之后，如果试图显示红色，则将显示灰色底纹。您必须用 RGB 颜色值来选择不在逻辑调色板中的颜色。

注意，GRAYS2 从不检查视频显示驱动程序是否支持调色板管理程序。在不支持调色板管理程序的视频模式（即所有非 256 种颜色的视频模式）下运行 GRAYS2 时，GRAYS2 的功能与 GRASY1 相同。

16.1.3　调色板消息

如果有多个 Windows 程序同时使用调色板管理器，则活动窗口将获得调色板的优先权。最近的活动窗口将获得第二个优先权，以此类推。无论新程序什么时候变为活动的，Windows 调色板管理器通常都重新组织系统调色板。

如果程序在逻辑调色板中指定一种颜色，该颜色又是 20 种保留颜色之一，那么 Windows 将把逻辑调色板条目映射给该颜色。另外，如果两个或多个应用程序都在它们的逻辑调色板中指定了同一种颜色，那么这些应用程序将共享系统调色板条目。程序可以通过将 PALETTEENTRY 结构的 peFlags 字段指定为常数 PC_NOCOLLAPSE 来忽

略该默认状态。(其余两个可能的标记是 PC_EXPLICIT(用于显示系统调色板)和 PC_RESERVED(用于调色板动画)。我将在本章的后面演示这两个标记。)

为了帮助组织系统调色板,Windows 调色板管理器需要包括两个发送给主窗口的消息。

第一个是 QM_QUERYNEWPALETTE。当主窗口活动时,该消息发送给主窗口。如果程序在您的窗口上绘画时使用了调色板管理器,则它必须处理该消息。GRAYS2 演示具体的方法。程序获得设备描述表句柄,并选进调色板,调用 RealizePalette,然后使窗口失效,以产生 WM_PAINT 消息。如果实现了逻辑调色板,则窗口过程从该消息返回 TRUE,否则返回 FALSE。

当系统调色板改成与 WM_QUERYNEWPALETTE 消息的结果相同时,Windows 将 WM_PALETTECHANGED 消息发送给由当前活动的窗口,来启动并终止处理窗口链的所有主窗口。这允许前台窗口有优先权。传递给窗口过程的 wParam 值是活动窗口的句柄。只有当 wParam 不等于程序的窗口句柄时,使用调色板管理器的程序才处理该消息。

通常情况下,当处理 WM_PALETTECHANGED 时,使用自定义调色板的任何程序都将调用 SelectPalette 和 RealizePalette。当后继的窗口在处理消息期间调用 RealizePalette 时,Windows 首先检查逻辑调色板中的 RGB 颜色是否与已加载到系统调色板中的 RGB 颜色相匹配。如果两个程序需要相同的颜色,那么这两个程序就共同使用一个系统调色板条目。接下来,Windows 将检查未使用的系统调色板条目。如果不存在没有使用的系统调色板条目,那么逻辑调色板中的颜色将从 20 种保留的条目映射到最接近的颜色。

如果您不关心程序非活动时客户区域的外观,那么您就不必处理 WM_PALETTECHANGED 消息。否则,您有两个选择。GRAYS2 演示其中之一:在处理 WM_QUERYNEWPALETTE 消息时,它获得设备描述表,选进调色板,然后调用 RealizePalette。这时可以在处理 WM_QUERYNEWPALETTE 时调用 InvalidateRect。而实际上,GRAYS2 调用 UpdateColors。这个函数通常比重新绘制窗口更有效,它改变窗口中像素的值来帮助保持以前的颜色。

使用调色板管理器的大多数程序都将具有 WM_QUERYNEWPALETTE 和 WM_PALETTECHANGED 消息处理,它们与 GRAYS2 中所示的方法完全相同。

16.1.4 调色板索引方法

程序 16-3 所示的 GRAYS3 程序与 GRAYS2 非常相似,只是在处理 WM_PAINT 期间使用了名为 PALETTEINDEX 的宏,而不是 PALETTERGB。

程序 16-3　GRAYS3 程序

GRAYS3.C

```
/*----------------------
   GRAYS3.C——Gray Shades Using Palette Manager
             (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Grays3") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;
     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Shades of Gray #3"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
```

```
    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HPALETTE  hPalette ;
    static int       cxClient, cyClient ;
    HBRUSH           hBrush ;
    HDC              hdc ;
    int              i ;
    LOGPALETTE *     plp ;
    PAINTSTRUCT      ps ;
    RECT             rect ;
    switch (message)
    {
    case WM_CREATE:
            // Set up a LOGPALETTE structure and create a palette

        plp = malloc (sizeof (LOGPALETTE) + 64 * sizeof (PALETTEENTRY)) ;

        plp -> palVersion    = 0x0300 ;
        plp -> palNumEntries = 65 ;

        for (i = 0 ; i < 65 ; i++)
        {
            plp -> palPalEntry[i].peRed   = (BYTE) min (255, 4 * i) ;
            plp -> palPalEntry[i].peGreen = (BYTE) min (255, 4 * i) ;
            plp -> palPalEntry[i].peBlue  = (BYTE) min (255, 4 * i) ;
            plp -> palPalEntry[i].peFlags = 0 ;
        }
        hPalette = CreatePalette (plp) ;
        free (plp) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

            // Select and realize the palette in the device context
```

```
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;

                // Draw the fountain of grays

            for (i = 0 ; i < 65 ; i++)
            {
                rect.left   = i * cxClient / 64 ;
                rect.top    = 0 ;
                rect.right  = (i + 1) * cxClient / 64 ;
                rect.bottom = cyClient ;

                hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;

                FillRect (hdc, &rect, hBrush) ;
                DeleteObject (hBrush) ;
            }

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_QUERYNEWPALETTE:
            if (!hPalette)
                return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, FALSE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

        case WM_PALETTECHANGED:
            if (!hPalette || (HWND) wParam == hwnd)
                break ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            UpdateColors (hdc) ;

            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
```

```
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

调色板索引的颜色不同于调色板 RGB 颜色,其高字节是 1,而低字节的值是当前在设备描述表中选择的逻辑调色板中的索引。在 GRAYS3 中,逻辑调色板有 65 个条目,用于这些条目的索引值为 0 ~ 64。值

PALETTEINDEX (0)

指黑色,

PALETTEINDEX (32)

指灰色,而

PALETTEINDEX (64)

指白色。

因为 Windows 不需要执行最近颜色的搜索,所以使用调色板索引比使用 RGB 值更有效。

16.1.5 查询调色板支持

可以容易地验证:当 Windows 在 16 位或 24 位视频模式下运行时,GRAYS2 和 GRAYS3 程序运行良好。但是在某些情况下,要使用调色板管理器的 Windows 应用程序可能要先确定设备驱动程序是否支持它。这时,您可以调用 GetDeviceCaps,并以视频显示的设备描述表句柄和 RASTERCAPS 作为参数。函数将返回由一系列标志组成的整数。通过在返回值和常量 RC_PALETTE 之间执行位与操作来检验支持的调色板:

```
RC_PALETTE & GetDeviceCaps (hdc, RASTERCAPS)
```

如果此值非 0,则视频显示器设备驱动程序将支持调色板操作。这种情况下,来自 GetDeviceCaps 的其他三个重要条目也可用的。函数调用

```
GetDeviceCaps (hdc, SIZEPALETTE)
```

将返回在显示卡上调色板表的总尺寸。这与同时显示的颜色总数相同。因为调色板管理器只用于每像素 8 位的视频显示模式,所以此值将是 256。

函数调用

```
GetDeviceCaps (hdc, NUMRESERVED)
```

返回在调色板表中的颜色数,该表是设备驱动程序为系统保留的。此值是 20。如果不

调用调色板管理器，Windows 应用程序在 256 色视频模式下只能使用这些纯色。要使用其余的 236 种颜色，程序必须使用调色板管理器函数。

还有一个附加条目：

```
GetDeviceCaps (hdc, COLORRES)
```

此值告诉您加载到硬件调色板表的 RGB 颜色值分辨率(以位为单位)。这些是进入数字模拟转换器的位。某些视频显示适配器只使用 6 位 ADC，所以该值是 18。有些适配器使用 8 位的 ADC，所以值是 24。

对于 Windows 程序而言，注意颜色分辨率并以此决定自己的行为是很有用的。例如，如果该颜色的分辨率为 18，那么程序试图申请到 128 种灰色是没什么意义的，因为只有 64 个离散的灰色可用。申请 128 种灰色将没有必要使用多余的条目来填充硬件调色板表。

16.1.6　系统调色板

在前面提到过，Windows 系统调色板直接与显示卡上的硬件调色板查询表相符。(然而，硬件调色板查询表可能比系统调色板的颜色分辨率低。)程序可以通过调用下面的函数来获得系统调色板中的某些或全部的 RGB 条目：

```
GetSystemPaletteEntries (hdc, uStart, uNum, &pe) ;
```

只有视频适配器模式支持调色板操作时，该函数才能运行。第二个和第三个参数是无符号整数，显示第一个调色板条目的索引和调色板条目数。最后一个参数是指向 PALETTEENTRY 类型的指针。

使用该函数有多种方式。程序可以定义一个如下的 PALETTEENTRY 结构：

```
PALETTEENTRY pe ;
```

然后可按下面的方法多次调用 GetSystemPaletteEntries：

```
GetSystemPaletteEntries (hdc, i, 1, &pe) ;
```

其中的 i 从 0 到某个值，该值小于从 GetDeviceCaps(带有 SIZEPALETTE 索引)返回的值(应是 255)。或者，程序要获得所有的系统调色板条目，可以通过定义指向 PALETTEENTRY 结构的指针，然后再分配足够的内存块，以保存与调色板大小指定同样多的 PALETTEENTRY 结构。

GetSystemPaletteEntries 函数确实允许您检验硬件调色板表。系统调色板中的条目按像素值增加的顺序排列，这些值用于表示视频显示缓冲区中的颜色。我将简单地讨论一下具体方法。

16.1.7 其他调色板函数

我们在前面看到过，Windows 程序能够改变系统调色板，但只是间接改变：第一步创建逻辑调色板，它基本上是程序要使用的 RGB 颜色值数组。CreatePalette 函数不会导致系统调色板或者显卡调色板索引表的任何变化。逻辑调色板必须在任何事情发生之前就选进设备描述表并实现。

程序可以通过调用

```
GetPaletteEntries (hPalette, uStart, uNum, &pe) ;
```

来查询逻辑调色板中的 RGB 颜色值。您可以按同样的方式来使用 GetSystemPaletteEntries 函数。但是要注意，第一个参数是逻辑调色板的句柄，而不是设备描述表的句柄。

创建逻辑调色板以后，允许您改变其中的值的相应函数是：

```
SetPaletteEntries (hPalette, uStart, uNum, &pe) ;
```

另外，记住调用此函数不引起系统调色板的任何变化——即使当前调色板选进了设备描述表。此函数也不改变逻辑调色板的尺寸。要改变逻辑调色板的尺寸，请使用 ResizePalette。

下面的函数接受 RGB 颜色引用值作为最后的参数，并将索引返回给逻辑调色板，该逻辑调色板与和它最接近的 RGB 颜色值相对应：

```
iIndex = GetNearestPaletteIndex (hPalette, cr) ;
```

第二个参数是 COLORREF 值。如果您愿意的话，调用 GetPaletteEntries 就可以获得逻辑调色板中实际的 RGB 颜色值。

如果程序在 8 位视频模式下需要多于 236 种自定义颜色，则可以调用 GetSystemPaletteUse。这允许程序设置 254 种自定义颜色，系统仅保留黑色和白色。不过，程序仅在最大化充满全屏幕时才应该这样，而且它还将一些系统颜色设为黑色和白色，以便标题栏和菜单等仍然可见。

16.1.8 光栅操作问题

从第五章可以了解到，GDI 允许使用不同的“绘画模式”或“光栅操作”来画线和填充区域。用 SetROP2 设置绘画模式，其中的“2”表示两个对象之间的二元光栅操作；三元光栅操作用于 BitBlt 及相关函数。这些光栅操作决定了正在画的对象像素与表面像素的结合方式。例如，您可以画一条直线，并让线上的像素与显示的像素按位异或的方式相结合。

光栅操作就是在像素位上逐位进行操作。改变调色板会影响到这些光栅操作。光栅操作的操作对象是像素位,而这些像素位可能与实际颜色没有联系。

通过运行 GRAYS2 或 GRAYS3 程序,您自己就可以得出这个结论。调整尺寸时,拖动顶部或底部的边界穿过窗口,Windows 利用反转背景像素位的光栅操作来显示拖动尺寸的边界,其目的是使拖动尺寸边界总是可见的。但在 GRAYS2 和 GRAYS3 程序中,您将看到各种随机改变的颜色,这些颜色恰好与对应于调色板表中未使用的条目,那是反转显示像素位的结果。可视颜色没有反转——只是像素位反转了。

正如在表 16-1 中所看到的一样,20 种标准保留颜色位于系统调色板的顶部和底部,以便光栅操作的结果仍然正常。然而,一旦您开始修改调色板——尤其是替换了保留颜色——那么颜色对象的光栅操作就变得没有意义了。

唯一保证的是光栅操作将用黑色和白色工作。黑色是系统调色板中的第一个条目(所有的像素位都设为 0),而白色是最后的条目(所有的像素位都设为 1)。这两个条目不能改变。如果需要预知在颜色对象上进行光栅操作的结果,则可以先获得系统调色板表,然后查看不同像素位值的 RGB 颜色值。

16.1.9　查看系统调色板

在 Windows 下运行的程序将处理逻辑调色板,为使逻辑调色板更好地服务于所有使用逻辑调色板的程序,Windows 将在系统调色板中设置颜色。该系统调色板把显示卡的硬件查找表镜像化。这样,查看系统调色板有助于调试调色板应用程序。

因为对于这个问题有三种截然不同的处理方式,所以我将演示三个程序,以显示系统调色板的内容。

SYSPAL1 程序,如程序 16-4 所示,使用了前面讲的 GetSystemPaletteEntries 函数。

程序 16-4　SYSPAL1 程序

SYSPAL1.C

```
/*--------------------
   SYSPAL1.C——Displays system palette
              (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
```

```
                PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;
     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("System Palette # 1"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
     if (!hwnd)
          return 0 ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
     HDC hdc ;
     int iPalSize ;
```

```
    hdc = GetDC (hwnd) ;
    iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
    ReleaseDC (hwnd, hdc) ;

    if (iPalSize != 256)
    {
        MessageBox (hwnd, TEXT ("This program requires that the video ")
                          TEXT ("display mode have a 256-color palette."),
                    szAppName, MB_ICONERROR) ;
        return FALSE ;
    }
    return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static int        cxClient, cyClient ;
    static SIZE       sizeChar ;
    HDC               hdc ;
    HPALETTE          hPalette ;
    int               i, x, y ;
    PAINTSTRUCT       ps ;
    PALETTEENTRY      pe [256] ;
    TCHAR             szBuffer [16] ;

    switch (message)
    {
    case WM_CREATE:
        if (!CheckDisplay (hwnd))
            return -1 ;

        hdc = GetDC (hwnd) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;
        GetTextExtentPoint32 (hdc, TEXT ("FF-FF-FF"), 10, &sizeChar) ;
        ReleaseDC (hwnd, hdc) ;
        return 0 ;

    case WM_DISPLAYCHANGE:
        if (!CheckDisplay (hwnd))
            DestroyWindow (hwnd) ;

        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;
```

```
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FIXED_FONT)) ;

        GetSystemPaletteEntries (hdc, 0, 256, pe) ;

        for (i = 0, x = 0, y = 0 ; i < 256 ; i++)
        {
            wsprintf (szBuffer, TEXT ("%02X-%02X-%02X"),
                      pe[i].peRed, pe[i].peGreen, pe[i].peBlue) ;
            TextOut (hdc, x, y, szBuffer, lstrlen (szBuffer)) ;

            if ((x += sizeChar.cx) + sizeChar.cx > cxClient)
            {
                x = 0 ;

                if ((y += sizeChar.cy) > cyClient)
                    break ;
            }
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_PALETTECHANGED:
        InvalidateRect (hwnd, NULL, FALSE) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

与 SYSPAL 系列中的其他程序一样，除非带有 SIZEPALETTE 参数的 GetDeviceCaps 返回值为 256，否则 SYSPAL1 不会运行。

注意，无论 SYSPAL1 的客户区域什么时候收到 WM_PALETTECHANGED 消息，它都是无效的。在合并 WM_PAINT 消息期间，SYSPAL1 调用 GetSystemPaletteEntries，并用一个含 256 个 PALETTEENTRY 结构的数组作为参数。RGB 值作为文本字符串显示在客户区。程序运行时，注意 20 种保留颜色是 RGB 值列表中的前 10 个和后 10 个，这与表 16-1 所示相同。

SYSPAL1 还是显示了一些有用的信息，但是并没有让您实际看到 256 种颜色。SYSPAL2 可以做到这一点，如程序 16-5 所示。

程序 16-5 SYSPAL2 程序

SYSPAL2.C

```
/* - - - - - - - - - - - - - - - - - - - -
    SYSPAL2.C——Displays system palette
                (c) Charles Petzold, 1998
   - - - - - - - - - - - - - - - - - - - - */

# include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal2") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    HWND        hwnd ;
    MSG         msg ;
    WNDCLASS    wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = NULL ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("System Palette #2"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;
    if (!hwnd)
```

```
        return 0 ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
    HDC hdc ;
    int iPalSize ;

    hdc = GetDC (hwnd) ;
    iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
    ReleaseDC (hwnd, hdc) ;

    if (iPalSize != 256)
    {
        MessageBox (hwnd, TEXT ("This program requires that the video ")
                          TEXT ("display mode have a 256-color palette."),
                    szAppName, MB_ICONERROR) ;
        return FALSE ;
    }
    return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HPALETTE  hPalette ;
    static int       cxClient, cyClient ;
    HBRUSH           hBrush ;
    HDC              hdc ;
    int              i, x, y ;
    LOGPALETTE *     plp ;
    PAINTSTRUCT      ps ;
    RECT             rect ;

    switch (message)
    {
    case WM_CREATE:
        if (!CheckDisplay (hwnd))
            return -1 ;
```

```
        plp = malloc (sizeof (LOGPALETTE) + 255 * sizeof (PALETTEENTRY)) ;

        plp -> palVersion    = 0x0300 ;
        plp -> palNumEntries = 256 ;

        for (i = 0 ; i < 256 ; i++ )
        {
             plp -> palPalEntry[i].peRed   = i ;
             plp -> palPalEntry[i].peGreen = 0 ;
             plp -> palPalEntry[i].peBlue  = 0 ;
             plp -> palPalEntry[i].peFlags = PC_EXPLICIT ;
        }

        hPalette = CreatePalette (plp) ;
        free (plp) ;
        return 0 ;

    case WM_DISPLAYCHANGE:
        if (!CheckDisplay (hwnd))
            DestroyWindow (hwnd) ;

        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;

        for (y = 0 ; y < 16 ; y++ )
        for (x = 0 ; x < 16 ; x++ )
        {
             hBrush = CreateSolidBrush (PALETTEINDEX (16 * y + x)) ;
             SetRect (&rect, x       * cxClient / 16, y       * cyClient / 16,
                             (x + 1) * cxClient / 16, (y + 1) * cyClient / 16);
             FillRect (hdc, &rect, hBrush) ;
             DeleteObject (hBrush) ;
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_PALETTECHANGED:
        if ((HWND) wParam != hwnd)
```

```
            InvalidateRect (hwnd, NULL, FALSE) ;
        return 0 ;

    case WM_DESTROY:
        DeleteObject (hPalette) ;
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SYSPAL2 在处理 WM_CREATE 消息期间创建了一个逻辑调色板。但是请注意，逻辑调色板中所有的 256 个值都是范围为 0～255 的调色板索引，并且 peFlags 字段是 PC_EXPLICIT。该标志是这样定义的："逻辑调色板条目低位字指定了一个硬件调色板索引。此标志允许应用程序给出显示设备调色板的内容。"由此可以看出，该标志特别倾向于我们要做的事情。

在处理 WM_PAINT 消息期间，SYSPAL2 将该调色板选进设备描述表并实现它。这不仅不会引起系统调色板的任何重组，而且允许程序使用 PALETTEINDEX 宏来指定系统调色板中的各种颜色。按照此方法，SYSPAL2 显示了 256 个矩形。另外，当您运行该程序时，请注意，顶行和底行的前 10 种颜色和后 10 种颜色正好是 20 种保留颜色，如 768 页表 16-1 所示。当您运行使用它们自己的逻辑调色板的程序时，则显示的内容就改变了。

如果您既喜欢看 SYSPAL2 中的颜色，又喜欢 RGB 的值，那么请与第八章的 WHATCLR 程序同时运行。

SYSPAL 系列中的第 3 版使用的技术对我来说是最近才发生的——其中的一些技术，从我开始研究 Windows 调色板管理器至今已经存在 7 年了。

事实上，所有的 GDI 函数都直接或间接地用 RGB 值指定颜色。在 GDI 内部，这将转换成与那个颜色相关的像素位。在某些视频模式中(例如，16 位或 24 位颜色模式)，这些转换是相当直接的。在其他视频模式中(4 位或 8 位颜色)，这可能涉及最近的颜色搜索。

然而，有两个 GDI 函数允许您直接指定像素位中的颜色。当然在这种方式中使用的这两个函数都具有高度的设备相关性。因为它们对设备的依赖性太强，以至于它们可以直接显示视频显示适配器上实际的调色板查找表。这两个函数是 BitBlt 和 StretchBlt。

程序 16-6 所示的 SYSPAL3 程序显示了使用 StretchBlt 显示系统调色板中颜色的方法。

程序 16-6　SYSPAL3 程序

SYSPAL3.C

```
/*-------------------------
    SYSPAL3.C——Displays system palette
                (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SysPal3") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;
     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("System Palette #3"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
     if (!hwnd)
         return 0 ;

     ShowWindow (hwnd, iCmdShow) ;
```

```
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
     HDC hdc ;
     int iPalSize ;

     hdc = GetDC (hwnd) ;

     iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
     ReleaseDC (hwnd, hdc) ;

     if (iPalSize != 256)
     {
          MessageBox (hwnd, TEXT ("This program requires that the video ")
                            TEXT ("display mode have a 256-color palette."),
                      szAppName, MB_ICONERROR) ;
          return FALSE ;
     }
     return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP  hBitmap ;
     static int      cxClient, cyClient ;
     BYTE            bits [256] ;
     HDC             hdc, hdcMem ;
     int             i ;
     PAINTSTRUCT     ps ;

     switch (message)
     {
     case WM_CREATE:
          if (!CheckDisplay (hwnd))
               return -1 ;

          for (i = 0 ; i < 256 ; i++)
               bits [i] = i ;

          hBitmap = CreateBitmap (16, 16, 1, 8, &bits) ;
          return 0 ;
```

```
        case WM_DISPLAYCHANGE:
            if (!CheckDisplay (hwnd))
                DestroyWindow (hwnd) ;

            return 0 ;

        case WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;

            StretchBlt (hdc,    0, 0, cxClient, cyClient,
                        hdcMem, 0, 0, 16, 16, SRCCOPY) ;

            DeleteDC (hdcMem) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            DeleteObject (hBitmap) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在 WM_CREATE 消息期间，SYSPAL3 使用 CreateBitmap 来创建 16 × 16 的每像素 8 位的位图。该函数的最后一个参数是包括数字 0～255 的 256 字节的数组。这些是 256 种可能的像素位值。在处理 WM_PAINT 消息的过程中，程序将这个位图选进内存设备描述表，用 StretchBlt 来显示并填充该客户区。Windows 仅将位图中的像素位传输到视频显示器硬件，从而允许这些像素位访问调色板查找表中的 256 个条目。接收 WM_PALETTECHANGED 消息时，程序的客户区可以不必置为无效——对查找表的任何修改都会立即反应到 SYSPAL3 的显示。

16.2　调色板动画

如果您以为本节标题中的“动画”一词指的是那些在屏幕上跑来跑去的“电脑宠物”，那您的眼光也太高了。虽然利用 Windows 调色板管理器确实可以做出一些动画，但那只

是一种有限的、特殊形式的动画。

通常,Windows 下的动画就是快速连续显示的一系列位图。调色板动画与这种方法有很大的区别。首先在屏幕上绘制出所有东西,然后操作调色板以改变这些东西的颜色,可能是画一些相对于屏幕背景来说是不可见的图像。您用这种方法就可以获得动画效果,而不必重画任何东西。因此调色板动画的速度是相当快的。

对于调色板动画,最初的创建工作与我们前面看见的有些不同:对于动画期间要修改的每种 RGB 颜色值,PALETTEENTRY 结构的 peFlags 字段必须设置为 PC_RESERVED。

通常,就像我们所看到的一样,在创建逻辑调色板时,将 peFlags 标记设为 0。这允许 GDI 将多个逻辑调色板中同样的颜色映射到相同的系统调色板条目。例如,假设两个 Windows 程序都创建了包含 RGB 条目 10-10-10 的逻辑调色板,那么在系统调色板表中,Windows 只需要一个 10-10-10 条目。但如果这两个程序中的一个使用调色板动画,那么您就不要再让 GDI 使用调色板了。调色板动画意味着速度非常快——而且如果不重画,那么它也只可能提高速度。当使用调色板动画的程序修改调色板时,它不会影响其他程序,或者迫使 GDI 重组系统调色板表。PC_RESERVED 的 peFlags 值为单个逻辑调色板保存系统调色板条目。

使用调色板动画时,通常可以在 WM_PAINT 消息期间调用 SelectPalette 和 RealizePalette,使用 PALETTEINDEX 宏来指定颜色。该宏将一个索引带进逻辑调色板表。

对于动画,您可能要通过改变调色板来响应 WM_TIMER 消息。要改变逻辑调色板中的 RGB 颜色值,请使用一个 PALETTEENTRY 结构的数组来调用函数 AnimatePalette。此函数速度很快,因为它只需要改变系统调色板以及显卡硬件调色板表中的条目。

16.2.1 跳动的球

程序 16-7 显示了 BOUNCE 程序的组件,但还有一个程序可显示跳动的球。为了简单起见,根据客户区的大小将球画成了椭圆形。因为本章有几个调色板动画程序,所以 PALANIM.C("调色板动画")文件包含一些通用内容。

程序 16-7 BOUNCE 程序

PALANIM.C

```
/*----------------------
   PALANIM.C——Palette Animation Shell Program
              (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>
```

```
extern HPALETTE CreateRoutine   (HWND) ;
extern void     PaintRoutine    (HDC, int, int) ;
extern void     TimerRoutine    (HDC, HPALETTE) ;
extern void     DestroyRoutine  (HWND, HPALETTE) ;

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

extern TCHAR szAppName [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS wndclass ;
     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, szTitle,
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
     if (!hwnd)
         return 0 ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
```

```
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

BOOL CheckDisplay (HWND hwnd)
{
    HDC hdc ;
    int iPalSize ;

    hdc = GetDC (hwnd) ;
    iPalSize = GetDeviceCaps (hdc, SIZEPALETTE) ;
    ReleaseDC (hwnd, hdc) ;

    if (iPalSize != 256)
    {
        MessageBox (hwnd, TEXT ("This program requires that the video ")
                          TEXT ("display mode have a 256-color palette."),
                    szAppName, MB_ICONERROR) ;
        return FALSE ;
    }
    return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HPALETTE  hPalette ;
    static int       cxClient, cyClient ;
    HDC              hdc ;
    PAINTSTRUCT      ps ;

    switch (message)
    {
    case WM_CREATE:
        if (!CheckDisplay (hwnd))
            return -1 ;

        hPalette = CreateRoutine (hwnd) ;
        return 0 ;

    case WM_DISPLAYCHANGE:
        if (!CheckDisplay (hwnd))
            DestroyWindow (hwnd) ;

        return 0 ;
```

```
    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;

        PaintRoutine (hdc, cxClient, cyClient) ;

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_TIMER:
        hdc = GetDC (hwnd) ;

        SelectPalette (hdc, hPalette, FALSE) ;

        TimerRoutine (hdc, hPalette) ;

        ReleaseDC (hwnd, hdc) ;
        return 0 ;

    case WM_QUERYNEWPALETTE:
        if (!hPalette)
            return FALSE ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        InvalidateRect (hwnd, NULL, TRUE) ;

        ReleaseDC (hwnd, hdc) ;
        return TRUE ;

    case WM_PALETTECHANGED:
        if (!hPalette || (HWND) wParam == hwnd)
            break ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        UpdateColors (hdc) ;

        ReleaseDC (hwnd, hdc) ;
```

```
            break ;

        case WM_DESTROY:
            DestroyRoutine (hwnd, hPalette) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

BOUNCE.C

```
/*------------------------
    BOUNCE.C——Palette Animation Demo
              (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>

#define ID_TIMER   1

TCHAR szAppName [] = TEXT ("Bounce") ;
TCHAR szTitle   [] = TEXT ("Bounce: Palette Animation Demo") ;

static LOGPALETTE * plp ;

HPALETTE CreateRoutine (HWND hwnd)
{
     HPALETTE hPalette ;
     int      i ;

     plp = malloc (sizeof (LOGPALETTE) + 33 * sizeof (PALETTEENTRY)) ;

     plp -> palVersion    = 0x0300 ;
     plp -> palNumEntries = 34 ;

     for (i = 0 ; i < 34 ; i++)
     {
          plp -> palPalEntry[i].peRed   = 255 ;
          plp -> palPalEntry[i].peGreen = (i == 0 ? 0 : 255) ;
          plp -> palPalEntry[i].peBlue  = (i == 0 ? 0 : 255) ;
          plp -> palPalEntry[i].peFlags = (i == 33 ? 0 : PC_RESERVED) ;
     }
     hPalette = CreatePalette (plp) ;

     SetTimer (hwnd, ID_TIMER, 50, NULL) ;
     return hPalette ;
}
```

```
void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
     HBRUSH hBrush ;
     int    i, x1, x2, y1, y2 ;
     RECT   rect ;
          // Draw window background using palette index 33

     SetRect (&rect, 0, 0, cxClient, cyClient) ;
     hBrush = CreateSolidBrush (PALETTEINDEX (33)) ;
     FillRect (hdc, &rect, hBrush) ;
     DeleteObject (hBrush) ;

          // Draw the 33 balls

     SelectObject (hdc, GetStockObject (NULL_PEN)) ;

     for (i = 0 ; i < 33 ; i++ )

     {
          x1 = i       * cxClient / 33 ;
          x2 = (i + 1) * cxClient / 33 ;

          if (i < 9)
          {
               y1 = i       * cyClient / 9 ;
               y2 = (i + 1) * cyClient / 9 ;
          }
          else if (i < 17)
          {
               y1 = (16 - i) * cyClient / 9 ;
               y2 = (17 - i) * cyClient / 9 ;
          }
          else if (i < 25)
          {
               y1 = (i - 16) * cyClient / 9 ;
               y2 = (i - 15) * cyClient / 9 ;
          }
          else
          {
               y1 = (32 - i) * cyClient / 9 ;
               y2 = (33 - i) * cyClient / 9 ;
          }

          hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;
          SelectObject (hdc, hBrush) ;
          Ellipse (hdc, x1, y1, x2, y2) ;
          DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRUSH))) ;
     }
```

```
    return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
    static BOOL bLeftToRight = TRUE ;
    static int  iBall ;

        // Set old ball to white

    plp -> palPalEntry[iBall].peGreen = 255 ;
    plp -> palPalEntry[iBall].peBlue  = 255 ;

    iBall += (bLeftToRight ? 1 : -1) ;

    if (iBall == (bLeftToRight ? 33 : -1))
    {
        iBall = (bLeftToRight ? 31 : 1) ;
        bLeftToRight ^= TRUE ;
    }

        // Set new ball to red

    plp -> palPalEntry[iBall].peGreen = 0 ;
    plp -> palPalEntry[iBall].peBlue  = 0 ;

        // Animate the palette

    AnimatePalette (hPalette, 0, 33, plp -> palPalEntry) ;
    return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
    KillTimer (hwnd, ID_TIMER) ;
    DeleteObject (hPalette) ;
    free (plp) ;
    return ;
}
```

除非 Windows 处于支持调色板的视频模式下，否则调色板动画将不能工作。因此，PALANIM.C 通过调用 CheckDisplay 函数（与 SYSPAL 程序中的函数相同）来开始 WM_CREATE 处理。

PALANIM.C 调用 BOUNCE.C 中的 4 个函数：在 WM_CREATE 消息期间调用 CreateRoutine（在 BOUNCE 中用于创建逻辑调色板）；在 WM_PAINT 消息期间调用 PaintRoutine；在 WM_TIMER 消息期间调用 TimerRoutine；在 WM_DESTROY 消息期间调用 De-

stroyRoutine(在 BOUNCE 中用于清除)。在调用 PaintRoutine 和 TimerRoutine 之前,PALANIM.C 获得设备描述表,并选进逻辑调色板。在调用 PaintRoutine 之前,它也实现调色板。PALANIM.C 期望 TimerRoutine 调用 AnimatePalette。尽管 AnimatePalette 需要从设备描述表中选择调色板,但它不需要调用 RealizePalette。

BOUNCE 中的球按"W"路线在客户区中来回跳动。客户区背景是白色,球是红色。任何时候,都可以在 33 个不重叠的位置之一看见球。这需要 34 个调色板条目:一个用于背景,其他 33 个用于不同位置的球。在 CreateRoutine 中,BOUNCE 初始化 PALETTEENTRY 结构的一个数组,将第一个调色板条目(与球在左上角的位置对应)设置为红色,其他的设置为白色。注意,对于除背景以外的所有条目,peFlags 字段都设置为 PC_RESERVED(背景是最后的一个调色板条目)。BOUNCE 通过将 Windows 定时器的间隔设置为 50 毫秒来终止 CreateRoutine。

BOUNCE 在 PaintRoutine 完成所有的绘画工作。窗口背景用一个实心画刷和调色板索引 33 所指定的颜色来绘制。33 个球的颜色是基于从 0 到 32 的调色板索引的颜色。当 BOUNCE 第一次在客户区内绘画时,调色板索引 0 映射成红色,其他调色板索引映射为白色。这将使球出现在左上角。

当 WndProc 处理 WM_TIMER 消息并调用 TimerRoutine 时,动画就发生了。TimerRoutine 通过调用 AnimatePalette 来结束,语法如下:

```
AnimatePalette (hPalette, uStart, uNum, &pe) ;
```

其中,第一个参数是调色板句柄,最后一个参数是指向数组的指针,该数组由一个或多个 PALETTEENTRY 结构组成。该函数改变逻辑调色板中从 uStart 条目到 uNum 条目之间的若干条目。逻辑调色板中新的 uStart 条目是 PALETTEENTRY 结构中的第一个成员。当心! uStart 参数是进入原始逻辑调色板表的索引,而不是进入 PALETTEENTRY 数组的索引。

为了方便起见,BOUNCE 使用 PALETTEENTRY 结构的数组,该结构是创建逻辑调色板时使用的 LOGPALETTE 结构的一部分。球的当前位置(从 0 到 32)保存在静态变量 iBall 中。在 TimerRoutine 期间,BOUNCE 将 PALETTEENTRY 成员设为白色,然后计算球的下一个位置,并将该元素设为红色。用下面的调用来改变调色板:

```
AnimatePalette (hPalette, 0, 33, plp -> palPalEntry) ;
```

GDI 改变前 33 个逻辑调色板条目(尽管实际上只改变了两个),使它与系统调色板表中的变化相对应,然后修改显示卡上的硬件调色板表。这样,不用重画球就开始移动了。

BOUNCE 运行时,您会发现同时运行 SYSPAL2 或 SYSPAL3 效果会更好。

尽管 AnimatePalette 运行得非常快,但是当只有一两个条目改变时,您还应该尽量避免改变所有的逻辑调色板条目。这在 BOUNCE 中有点复杂,因为球要来回地跳——

iBall 要先增加，然后再减少。一种方法是使用两个变量：iBallOld（设置球的当前位置）和 iBallMin（iBall 和 iBallOld 中较小的），然后就可以像下面这样调用 AnimatePalette 来只改变两个条目了：

```
iBallMin = min (iBall, iBallOld) ;
    AnimatePalette (hPal, iBallMin, 2, plp->palPalEntry + iBallMin) ;
```

还有另一种方法：我们先假定定义了一个 PALETTEENTRY 结构：

```
PALETTEENTRY pe ;
```

在 TimerRoutine 期间，将 PALETTEENTRY 字段置为白色，并调用 AnimatePalette 来改变逻辑调色板中 iBall 位置的一个条目：

```
pe.peRed   = 255 ;
pe.peGreen = 255 ;
pe.peBlue  = 255 ;
pe.peFlags = PC_RESERVED ;
AnimatePalette (hPalette, iBall, 1, &pe) ;
```

然后计算显示在 BOUNCE 中的 iBall 的新值，将 PALETTEENTRY 结构的字段定义为红色，然后再次调用 AnimatePalette：

```
pe.peRed   = 255 ;
pe.peGreen = 0 ;
pe.peBlue  = 0 ;
pe.peFlags = PC_RESERVED ;
AnimatePalette (hPalette, iBall, 1, &pe) ;
```

尽管跳动的球是对动画的一个传统的简单说明，但它实际上并不适合调色板动画，因为必须先画出球的所有可能位置。调色板动画更适合于显示运动的重复图案。

16.2.2　一个条目的调色板动画

调色板动画中一个更有趣的方面就是，可以只使用一个调色板条目来完成一些有趣的技术。例如程序 16-8 所示的 FADER 程序。这个程序也需要前面的 PALANIM.C 文件。

程序 16-8　FADER 程序

FADER.C

```
/*-------------------
   FADER.C——Palette Animation Demo
            (c) Charles Petzold, 1998
   -------------------*/
```

```
# include < windows.h >

# define ID_TIMER 1

TCHAR szAppName [ ] = TEXT ("Fader") ;
TCHAR szTitle    [ ] = TEXT ("Fader: Palette Animation Demo") ;

static LOGPALETTE lp ;

HPALETTE CreateRoutine (HWND hwnd)
{
     HPALETTE hPalette ;

     lp.palVersion                = 0x0300 ;
     lp.palNumEntries             = 1 ;
     lp.palPalEntry[0].peRed      = 255 ;
     lp.palPalEntry[0].peGreen    = 255 ;
     lp.palPalEntry[0].peBlue     = 255 ;
     lp.palPalEntry[0].peFlags    = PC_RESERVED ;

     hPalette = CreatePalette (&lp) ;

     SetTimer (hwnd, ID_TIMER, 50, NULL) ;
     return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
     static TCHAR  szText [ ] = TEXT (" Fade In and Out ") ;
     int           x, y ;
     SIZE          sizeText ;

     SetTextColor (hdc, PALETTEINDEX (0)) ;
     GetTextExtentPoint32 (hdc, szText, lstrlen (szText), &sizeText) ;

     for (x = 0 ; x < cxClient ; x += sizeText.cx)
     for (y = 0 ; y < cyClient ; y += sizeText.cy)
     {
          TextOut (hdc, x, y, szText, lstrlen (szText)) ;
     }

     return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
     static BOOL bFadeIn = TRUE ;
```

```
    if (bFadeIn)
    {
        lp.palPalEntry[0].peRed   -= 4 ;
        lp.palPalEntry[0].peGreen -= 4 ;

        if (lp.palPalEntry[0].peRed == 3)
            bFadeIn = FALSE ;
    }
    else
    {
        lp.palPalEntry[0].peRed   += 4 ;
        lp.palPalEntry[0].peGreen += 4 ;

        if (lp.palPalEntry[0].peRed == 255)
            bFadeIn = TRUE ;
    }

    AnimatePalette (hPalette, 0, 1, lp.palPalEntry) ;
    return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
    KillTimer (hwnd, ID_TIMER) ;
    DeleteObject (hPalette) ;
    return ;
}
```

FADER 把文本字符串"Fade In And Out"满满地显示在客户区上。文本首先显示为白色，这对于白色背景的窗口来说是看不出来的。通过使用调色板动画，FADER 慢慢地将文本的颜色改为蓝色，然后再改回白色，这样一遍一遍地重复。文本就有时隐时现的显示效果了。

FADER 用 CreateRoutine 函数创建了逻辑调色板，它只需要一个调色板条目，并将颜色初始化为白色——红色、绿色和蓝色值都设为 255。在 PaintRoutine 中（您可能想起，当逻辑调色板选进设备描述表并实现以后，PALANIM 调用过此函数），FADER 调用 SetTextColor 将文本颜色设置为 PALETTEINDEX(0)。这意味着文本颜色设置为调色板表格中的第一个条目，此条目初始为白色，然后 FADER 用"Fade In And Out"文本字符串填充客户区。这时，窗口背景是白色，文本也是白色，所以文本不可见。

在 TimerRoutine 函数中，FADER 通过改变 PALETTEENTRY 结构并将其传递给 AnimatePalette 来完成调色板动画。最初，对每一个 WM_TIMER 消息，程序都将红色和绿色值减 4，直到等于 3，然后将这些值加 4，直到等于 255。这将使文本颜色逐渐从白色变到蓝色，然后又回到白色。

程序 16-9 所示的 ALLCOLOR 程序只用了逻辑调色板的一个条目来显示视频适配

器可以着色的所有颜色。当然,程序不是同时显示这些颜色,而是连续显示。如果视频适配器有 18 位的分辨率(这时能有 262 144 种不同的颜色),那么在两种颜色间隔 55 毫秒的速度下,需要 4 小时才能在屏幕上看到所有的颜色!

程序 16-9　ALLCOLOR 程序

ALLCOLOR.C

```
/*-------------------
    ALLCOLOR.C——Palette Animation Demo
                (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

#define ID_TIMER    1

TCHAR szAppName [] = TEXT ("AllColor") ;
TCHAR szTitle     [] = TEXT ("AllColor: Palette Animation Demo") ;

static int              iIncr ;
static PALETTEENTRY pe ;

HPALETTE CreateRoutine (HWND hwnd)
{
     HDC          hdc ;
     HPALETTE     hPalette ;
     LOGPALETTE   lp ;
          // Determine the color resolution and set iIncr
     hdc = GetDC (hwnd) ;
     iIncr = 1 << (8 - GetDeviceCaps (hdc, COLORRES) / 3) ;
     ReleaseDC (hwnd, hdc) ;

          // Create the logical palette

     lp.palVersion                = 0x0300 ;
     lp.palNumEntries             = 1 ;
     lp.palPalEntry[0].peRed      = 0 ;
     lp.palPalEntry[0].peGreen    = 0 ;
     lp.palPalEntry[0].peBlue     = 0 ;
     lp.palPalEntry[0].peFlags    = PC_RESERVED ;

     hPalette = CreatePalette (&lp) ;

          // Save global for less typing

     pe = lp.palPalEntry[0] ;
```

```
    SetTimer (hwnd, ID_TIMER, 10, NULL) ;
    return hPalette ;
}

void DisplayRGB (HDC hdc, PALETTEENTRY * ppe)
{
    TCHAR szBuffer [16] ;

    wsprintf (szBuffer, TEXT (" %02X-%02X-%02X "),
              ppe -> peRed, ppe -> peGreen, ppe -> peBlue) ;

    TextOut (hdc, 0, 0, szBuffer, lstrlen (szBuffer)) ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
    HBRUSH    hBrush ;
    RECT      rect ;

        // Draw Palette Index 0 on entire window

    hBrush = CreateSolidBrush (PALETTEINDEX (0)) ;
    SetRect (&rect, 0, 0, cxClient, cyClient) ;
    FillRect (hdc, &rect, hBrush) ;
    DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRUSH))) ;
        // Display the RGB value

    DisplayRGB (hdc, &pe) ;
    return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
    static BOOL bRedUp = TRUE, bGreenUp = TRUE, bBlueUp = TRUE ;

        // Define new color value

    pe.peBlue += (bBlueUp ? iIncr : -iIncr) ;

    if (pe.peBlue == (BYTE) (bBlueUp ? 0 : 256 - iIncr))
    {
        pe.peBlue = (bBlueUp ? 256 - iIncr : 0) ;
        bBlueUp ^= TRUE ;
        pe.peGreen += (bGreenUp ? iIncr : -iIncr) ;

        if (pe.peGreen == (BYTE) (bGreenUp ? 0 : 256 - iIncr))
        {
```

```
                pe.peGreen = (bGreenUp ? 256 - iIncr : 0) ;
                bGreenUp ^= TRUE ;
                pe.peRed += (bRedUp ? iIncr : -iIncr) ;

                if (pe.peRed == (BYTE) (bRedUp ? 0 : 256 - iIncr))
                {
                    pe.peRed = (bRedUp ? 256 - iIncr : 0) ;
                    bRedUp ^= TRUE ;
                }
            }
        }
        // Animate the palette

        AnimatePalette (hPalette, 0, 1, &pe) ;
        DisplayRGB (hdc, &pe) ;
        return ;
    }

    void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
    {
        KillTimer (hwnd, ID_TIMER) ;
        DeleteObject (hPalette) ;
        return ;
    }
```

在结构上，ALLCOLOR 与 FADER 非常相似。在 CreateRoutine 中，ALLCOLOR 只用一个设为黑色的调色板条目(PALETTEENTRY 结构的 red、green 和 blue 字段设为 0)来创建调色板。在 PaintRoutine 中，ALLCOLOR 用 PALETTEINDEX(0)创建实心画刷，并调用 FillRect 来用此画刷为整个客户区着色。

在 TimerRoutine 中，ALLCOLOR 通过改变 PALETTEENTRY 颜色并调用 AnimatePalette 来激活调色板。我编写 ALLCOLOR 程序，以便颜色光滑变化。首先，蓝色值渐渐增加。达到最大时，绿色值增加，而蓝色值渐渐减少。红色、绿色和蓝色值的增加和减少取决于 iIncr 变量。在 CreateRoutine 期间，这将根据用 COLORRES 参数从 GetDeviceCaps 返回的值来计算。例如，如果 GetDeviceCaps 返回 18，那么 iIncr 设为 4——获得所有颜色所需要的最小值。

ALLCOLOR 还在客户区的左上角显示当前的 RGB 颜色值。我最初添加这个代码是出于测试目的，但是现在证明它是有用的，所以我保留了它。

16.2.3　工程应用程序

在工程应用程序中，动画对于显示机械或电的过程很有用。它不仅可以在计算机屏幕上显示内燃机，而且动画的确可以使它变得更加生动，更清楚地显示其工作过程。

使用调色板动画的一个好的示例就是显示流体通过管子的过程。这是一个例子，

图像不必十分精确——实际上，如果图像很精确(就像看透明的管子)，则很难说明管子里的流体是如何运动的。这时用符号会更好一些。程序 16-10 所示的 PIPES 程序是此技术的简单示范：在客户区有两个水平的管子，流体在上面的管子里从左向右运动，而在下面的管子里从右向左运动。

程序 16-10　PIPES 程序

PIPES.C

```
/*------------------
    PIPES.C——Palette Animation Demo
              (c) Charles Petzold, 1998
   ------------------*/

#include <windows.h>

#define ID_TIMER   1
TCHAR szAppName [] = TEXT ("Pipes") ;
TCHAR szTitle   [] = TEXT ("Pipes: Palette Animation Demo") ;

static LOGPALETTE * plp ;

HPALETTE CreateRoutine (HWND hwnd)
{
     HPALETTE  hPalette ;
     int       i ;

     plp = malloc (sizeof (LOGPALETTE) + 32 * sizeof (PALETTEENTRY)) ;

          // Initialize the fields of the LOGPALETTE structure

     plp->palVersion    = 0x300 ;
     plp->palNumEntries = 16 ;

     for (i = 0 ; i <= 8 ; i++)
     {
          plp->palPalEntry[i].peRed   = (BYTE) min (255, 0x20 * i) ;
          plp->palPalEntry[i].peGreen = 0 ;
          plp->palPalEntry[i].peBlue  = (BYTE) min (255, 0x20 * i) ;
          plp->palPalEntry[i].peFlags = PC_RESERVED ;

          plp->palPalEntry[16 - i] = plp->palPalEntry[i] ;
          plp->palPalEntry[16 + i] = plp->palPalEntry[i] ;
          plp->palPalEntry[32 - i] = plp->palPalEntry[i] ;
     }

     hPalette = CreatePalette (plp) ;
```

```
    SetTimer (hwnd, ID_TIMER, 100, NULL) ;
    return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
    HBRUSH hBrush ;
    int    i ;
    RECT   rect ;

        // Draw window background

    SetRect (&rect, 0, 0, cxClient, cyClient) ;
    hBrush = SelectObject (hdc, GetStockObject (WHITE_BRUSH)) ;
    FillRect (hdc, &rect, hBrush) ;

        // Draw the interiors of the pipes

    for (i = 0 ; i < 128 ; i++)
    {
        hBrush = CreateSolidBrush (PALETTEINDEX (i % 16)) ;
        SelectObject (hdc, hBrush) ;

        rect.left   = (127 - i) * cxClient / 128 ;
        rect.right  = (128 - i) * cxClient / 128 ;
        rect.top    = 4 * cyClient / 14 ;
        rect.bottom = 5 * cyClient / 14 ;

        FillRect (hdc, &rect, hBrush) ;

        rect.left   =  i    * cxClient / 128 ;
        rect.right  = (i + 1) * cxClient / 128 ;
        rect.top    =  9 * cyClient / 14 ;
        rect.bottom = 10 * cyClient / 14 ;
        FillRect (hdc, &rect, hBrush) ;

        DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRUSH))) ;
    }
        // Draw the edges of the pipes

    MoveToEx (hdc, 0,      4 * cyClient / 14, NULL) ;
    LineTo   (hdc, cxClient, 4 * cyClient / 14) ;

    MoveToEx (hdc, 0,      5 * cyClient / 14, NULL) ;
    LineTo   (hdc, cxClient, 5 * cyClient / 14) ;

    MoveToEx (hdc, 0,      9 * cyClient / 14, NULL) ;
```

```
    LineTo    (hdc, cxClient, 9 * cyClient / 14) ;

    MoveToEx (hdc, 0,       10 * cyClient / 14, NULL) ;
    LineTo    (hdc, cxClient, 10 * cyClient / 14) ;
    return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
    static int iIndex ;

    AnimatePalette (hPalette, 0, 16, plp -> palPalEntry + iIndex) ;
    iIndex = (iIndex + 1) % 16 ;

    return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
    KillTimer (hwnd, ID_TIMER) ;
    DeleteObject (hPalette) ;
    free (plp) ;
    return ;
}
```

PIPES 为动画使用了 16 个调色板条目，而您可能会使用更少的条目。最小化时，真正需要的是有足够的条目来显示流动的方向。用三个调色板条目要比用一个静态箭头好。

程序 16-11 所示的 TUNNEL 程序是这批程序中最贪心的程序，它为动画使用了 128 个调色板条目，但是从效果来看值得这样做。

程序 16-11　TUNNEL 程序

TUNNEL.C

```
/*------------------------
    TUNNEL.C——Palette Animation Demo
              (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>

#define ID_TIMER  1

TCHAR szAppName [] = TEXT ("Tunnel") ;
TCHAR szTitle   [] = TEXT ("Tunnel: Palette Animation Demo") ;
```

```
static LOGPALETTE * plp ;

HPALETTE CreateRoutine (HWND hwnd)
{
     BYTE        byGrayLevel ;
     HPALETTE    hPalette ;
     int         i ;

     plp = malloc (sizeof (LOGPALETTE) + 255 * sizeof (PALETTEENTRY)) ;
          // Initialize the fields of the LOGPALETTE structure

     plp -> palVersion    = 0x0300 ;
     plp -> palNumEntries = 128 ;

     for (i = 0 ; i < 128 ; i++)
     {
          if (i < 64)
             byGrayLevel = (BYTE) (4 * i) ;
          else
             byGrayLevel = (BYTE) min (255, 4 * (128 - i)) ;

          plp -> palPalEntry[i].peRed   = byGrayLevel ;
          plp -> palPalEntry[i].peGreen = byGrayLevel ;
          plp -> palPalEntry[i].peBlue  = byGrayLevel ;
          plp -> palPalEntry[i].peFlags = PC_RESERVED ;

          plp -> palPalEntry[i + 128].peRed   = byGrayLevel ;
          plp -> palPalEntry[i + 128].peGreen = byGrayLevel ;
          plp -> palPalEntry[i + 128].peBlue  = byGrayLevel ;
          plp -> palPalEntry[i + 128].peFlags = PC_RESERVED ;
     }

     hPalette = CreatePalette (plp) ;

     SetTimer (hwnd, ID_TIMER, 50, NULL) ;
     return hPalette ;
}

void PaintRoutine (HDC hdc, int cxClient, int cyClient)
{
     HBRUSH  hBrush ;
     int     i ;
     RECT    rect ;

     for (i = 0 ; i < 127 ; i++)
     {
               // Use a RECT structure for each of 128 rectangles
```

```
            rect.left   =              i * cxClient / 255 ;
            rect.top    =              i * cyClient / 255 ;
            rect.right  = cxClient -   i * cxClient / 255 ;
            rect.bottom = cyClient -   i * cyClient / 255 ;

            hBrush = CreateSolidBrush (PALETTEINDEX (i)) ;
                // Fill the rectangle and delete the brush

            FillRect (hdc, &rect, hBrush) ;
            DeleteObject (hBrush) ;
        }
        return ;
}

void TimerRoutine (HDC hdc, HPALETTE hPalette)
{
        static int iLevel ;
        iLevel = (iLevel + 1) % 128 ;

        AnimatePalette (hPalette, 0, 128, plp -> palPalEntry + iLevel) ;
        return ;
}

void DestroyRoutine (HWND hwnd, HPALETTE hPalette)
{
        KillTimer (hwnd, ID_TIMER) ;
        DeleteObject (hPalette) ;
        free (plp) ;
        return ;
}
```

TUNNEL 在 128 个调色板条目中使用 64 种移动的灰色底纹——从黑到白，再从白到黑——给出在隧道旅行的效果。

16.3　调色板和真实世界图像

当然，尽管我们已经完成了许多有趣的事：连续显示色彩的底纹，做了调色板动画，但调色板管理器的真正目的是允许在 8 位视频模式下显示真实世界中的图像。对于本章的其余部分，我们正好研究一下。正如您所期望的，在使用紧缩 DIB、GDI 位图对象和 DIB 块时，必须按不同的方法来使用调色板。下面的 6 个程序阐明了用调色板来处理位图的各种技术。

16.3.1　调色板和紧缩 DIB

下面三个程序，有助于我们建立处理紧缩 DIB 内存块的一系列函数。这些函数都在程

序16-12所示的 PACKEDIB 文件中。

程序 16-12　PACKEDIB 文件

PACKEDIB.H

```
/*-------------------------
    PACKEDIB.H ——Header file for PACKEDIB.C
                 (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

BITMAPINFO * PackedDibLoad (PTSTR szFileName) ;
int PackedDibGetWidth (BITMAPINFO * pPackedDib) ;
int PackedDibGetHeight (BITMAPINFO * pPackedDib) ;
int PackedDibGetBitCount (BITMAPINFO * pPackedDib) ;
int PackedDibGetRowLength (BITMAPINFO * pPackedDib) ;
int PackedDibGetInfoHeaderSize (BITMAPINFO * pPackedDib) ;
int PackedDibGetColorsUsed (BITMAPINFO * pPackedDib) ;
int PackedDibGetNumColors (BITMAPINFO * pPackedDib) ;
int PackedDibGetColorTableSize (BITMAPINFO * pPackedDib) ;
RGBQUAD * PackedDibGetColorTablePtr (BITMAPINFO * pPackedDib) ;
RGBQUAD * PackedDibGetColorTableEntry (BITMAPINFO * pPackedDib, int i) ;
BYTE * PackedDibGetBitsPtr (BITMAPINFO * pPackedDib) ;
int PackedDibGetBitsSize (BITMAPINFO * pPackedDib) ;
HPALETTE PackedDibCreatePalette (BITMAPINFO * pPackedDib) ;
```

PACKEDIB.C

```
/*-------------------------
    PACKEDIB.C——Routines for using packed DIBs
                (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

/*-------------------------------
    PackedDibLoad: Load DIB File as Packed-Dib Memory Block
   -------------------------------*/

BITMAPINFO * PackedDibLoad (PTSTR szFileName)
{
    BITMAPFILEHEADER  bmfh ;
    BITMAPINFO      * pbmi ;
    BOOL              bSuccess ;
    DWORD             dwPackedDibSize, dwBytesRead ;
    HANDLE            hFile ;
        // Open the file: read access, prohibit write access
```

```
    hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                        OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
    if (hFile == INVALID_HANDLE_VALUE)
        return NULL ;

        // Read in the BITMAPFILEHEADER

    bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                         &dwBytesRead, NULL) ;

    if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEADER))
                  || (bmfh.bfType != * (WORD *) "BM"))
    {
         CloseHandle (hFile) ;
         return NULL ;
    }

        // Allocate memory for the packed DIB & read it in

    dwPackedDibSize = bmfh.bfSize - sizeof (BITMAPFILEHEADER) ;

    pbmi = malloc (dwPackedDibSize) ;

    bSuccess = ReadFile (hFile, pbmi, dwPackedDibSize, &dwBytesRead, NULL) ;
    CloseHandle (hFile) ;

    if (!bSuccess || (dwBytesRead != dwPackedDibSize))
    {
         free (pbmi) ;
         return NULL ;
    }

    return pbmi ;
}

/*-----------------------
    Functions to get information from packed DIB
  -----------------------*/

int PackedDibGetWidth (BITMAPINFO * pPackedDib)
{
    if (pPackedDib->bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
        return ((PBITMAPCOREINFO)pPackedDib)->bmciHeader.bcWidth ;
    else
        return pPackedDib->bmiHeader.biWidth ;
}

int PackedDibGetHeight (BITMAPINFO * pPackedDib)
```

```
{
     if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
          return ((PBITMAPCOREINFO)pPackedDib) -> bmciHeader.bcHeight ;
     else
          return abs (pPackedDib -> bmiHeader.biHeight) ;
}

int PackedDibGetBitCount (BITMAPINFO * pPackedDib)
{
     if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
          return ((PBITMAPCOREINFO)pPackedDib) -> bmciHeader.bcBitCount ;
     else
          return pPackedDib -> bmiHeader.biBitCount ;
}

int PackedDibGetRowLength (BITMAPINFO * pPackedDib)
{
     return ((PackedDibGetWidth (pPackedDib) *
               PackedDibGetBitCount (pPackedDib) + 31) & ~31) >> 3 ;
}

/*-------------------------
     PackedDibGetInfoHeaderSize includes possible color masks!
   -------------------------*/

int PackedDibGetInfoHeaderSize (BITMAPINFO * pPackedDib)
{
     if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
          return ((PBITMAPCOREINFO)pPackedDib) -> bmciHeader.bcSize ;

     else if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPINFOHEADER))
          return pPackedDib -> bmiHeader.biSize +
                    (pPackedDib -> bmiHeader.biCompression ==
                                        BI_BITFIELDS ? 12 : 0) ;
     else return pPackedDib -> bmiHeader.biSize ;
}

/*-------------------------
     PackedDibGetColorsUsed returns value in information header;
          could be 0 to indicate non-truncated color table!
   -------------------------*/

int PackedDibGetColorsUsed (BITMAPINFO * pPackedDib)
{
     if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
          return 0 ;
     else
          return pPackedDib -> bmiHeader.biClrUsed ;
```

```
}

/*----------------------------------------
   PackedDibGetNumColors is actual number of entries in color table
  ----------------------------------------*/

int PackedDibGetNumColors (BITMAPINFO * pPackedDib)
{
     int iNumColors ;

     iNumColors = PackedDibGetColorsUsed (pPackedDib) ;

     if (iNumColors == 0 && PackedDibGetBitCount (pPackedDib) < 16)
          iNumColors = 1 << PackedDibGetBitCount (pPackedDib) ;
     return iNumColors ;
}

int PackedDibGetColorTableSize (BITMAPINFO * pPackedDib)
{
     if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
          return PackedDibGetNumColors (pPackedDib) * sizeof (RGBTRIPLE) ;
     else
          return PackedDibGetNumColors (pPackedDib) * sizeof (RGBQUAD) ;
}

RGBQUAD * PackedDibGetColorTablePtr (BITMAPINFO * pPackedDib)
{
     if (PackedDibGetNumColors (pPackedDib) == 0)
          return 0 ;
     return (RGBQUAD * ) (((BYTE * ) pPackedDib) +
                                   PackedDibGetInfoHeaderSize (pPackedDib)) ;
}

RGBQUAD * PackedDibGetColorTableEntry (BITMAPINFO * pPackedDib, int i)
{
     if (PackedDibGetNumColors (pPackedDib) == 0)
          return 0 ;

     if (pPackedDib -> bmiHeader.biSize == sizeof (BITMAPCOREHEADER))
          return (RGBQUAD * )
               (((RGBTRIPLE * ) PackedDibGetColorTablePtr (pPackedDib)) + i) ;
     else
          return PackedDibGetColorTablePtr (pPackedDib) + i ;
}
/*----------------------------
   PackedDibGetBitsPtr finally!
  ----------------------------*/

BYTE * PackedDibGetBitsPtr (BITMAPINFO * pPackedDib)
```

```
{
     return ((BYTE * ) pPackedDib) + PackedDibGetInfoHeaderSize (pPackedDib) +
                                     PackedDibGetColorTableSize (pPackedDib) ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     PackedDibGetBitsSize can be calculated from the height and row length
          if it's not explicitly in the biSizeImage field
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

int PackedDibGetBitsSize (BITMAPINFO * pPackedDib)
{
     if ((pPackedDib -> bmiHeader.biSize != sizeof (BITMAPCOREHEADER)) &&
        (pPackedDib -> bmiHeader.biSizeImage != 0))
        return pPackedDib -> bmiHeader.biSizeImage ;

     return PackedDibGetHeight (pPackedDib) *
          PackedDibGetRowLength (pPackedDib) ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - -
     PackedDibCreatePalette creates logical palette from PackedDib
   - - - - - - - - - - - - - - - - - - - - - - - - - - - */

HPALETTE PackedDibCreatePalette (BITMAPINFO * pPackedDib)
{
     HPALETTE        hPalette ;
     int              i, iNumColors ;
     LOGPALETTE * plp ;
     RGBQUAD     * prgb ;

     if (0 == (iNumColors = PackedDibGetNumColors (pPackedDib)))
          return NULL ;

     plp = malloc (sizeof (LOGPALETTE) *
                    (iNumColors - 1) * sizeof (PALETTEENTRY)) ;

     plp -> palVersion     = 0x0300 ;
     plp -> palNumEntries = iNumColors ;

     for (i = 0 ; i < iNumColors ; i++)
     {
          prgb = PackedDibGetColorTableEntry (pPackedDib, i) ;

          plp -> palPalEntry[i].peRed    = prgb -> rgbRed ;
          plp -> palPalEntry[i].peGreen  = prgb -> rgbGreen ;
          plp -> palPalEntry[i].peBlue   = prgb -> rgbBlue ;
          plp -> palPalEntry[i].peFlags  = 0 ;
     }
```

```
    hPalette = CreatePalette (plp) ;
    free (plp) ;

    return hPalette ;
}
```

第一个函数是 PackedDibLoad,它唯一的参数是文件名,并返回指向内存中紧缩 DIB 的指针。其他所有函数都将这个紧缩缩 DIB 指针作为它们的第一个参数,并返回有关 DIB 的信息。这些函数按"从下向上"顺序排列到文件中。每个函数都使用从前面函数获得的信息。

我不倾向于说这是在处理紧缩 DIB 时有用的"完整"函数集。而且,我也不想汇编一个真正的扩展集,因为我不认为这是处理紧缩 DIB 的一个好方法。在写类似下面的函数时,您会很明显地发现这一点:

```
dwPixel = PackedDibGetPixel (pPackedDib, x, y) ;
```

这种函数包括太多的嵌套函数调用,以致于效率非常低,而且很慢。本章的后面将讨论一种我认为更好的方法。

另外,您将注意到,其中许多函数都需要对 OS/2 兼容的 DIB 采取不同的处理过程。这样,函数将频繁地检查 BITMAPINFO 结构的第一个字段是否与 BITMAPCOREHEADER 结构的大小相同。

此处特别有意义的是最后一个名为 PackedDibCreatePalette 的函数。这个函数用 DIB 中的颜色表来创建调色板。如果 DIB 中没有颜色表(这意味着 DIB 的每像素有 16、24 或 32 位),那么就不创建调色板。我们有时将从 DIB 颜色表创建的调色板称为 DIB 自己的调色板。

PACKEDIB 文件都放在 SHOWDIB3,如程序 16-13 所示。

程序 16-13　SHOWDIB3 程序

SHOWDIB3.C

```
/*------------------------
   SHOWDIB3.C——Displays DIB with native palette
               (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include "PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
```

```
TCHAR szAppName[ ] = TEXT ("ShowDib3") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;
     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #3: Native Palette"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BITMAPINFO * pPackedDib ;
     static HPALETTE     hPalette ;
```

```
static int              cxClient, cyClient ;
static OPENFILENAME     ofn ;
static TCHAR            szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
static TCHAR            szFilter[ ] = TEXT ("Bitmap Files ( * .BMP) \ 0 * .bmp \ 0")
                                      TEXT ("All Files ( * . * ) \ 0 * . * \ 0 \ 0") ;
HDC                     hdc ;
PAINTSTRUCT             ps ;
switch (message)
{
case WM_CREATE:
     ofn.lStructSize            = sizeof (OPENFILENAME) ;
     ofn.hwndOwner              = hwnd ;
     ofn.hInstance              = NULL ;
     ofn.lpstrFilter            = szFilter ;
     ofn.lpstrCustomFilter      = NULL ;
     ofn.nMaxCustFilter         = 0 ;
     ofn.nFilterIndex           = 0 ;
     ofn.lpstrFile              = szFileName ;
     ofn.nMaxFile               = MAX_PATH ;
     ofn.lpstrFileTitle         = szTitleName ;
     ofn.nMaxFileTitle          = MAX_PATH ;
     ofn.lpstrInitialDir        = NULL ;
     ofn.lpstrTitle             = NULL ;
     ofn.Flags                  = 0 ;
     ofn.nFileOffset            = 0 ;
     ofn.nFileExtension         = 0 ;
     ofn.lpstrDefExt            = TEXT ("bmp") ;
     ofn.lCustData              = 0 ;
     ofn.lpfnHook               = NULL ;
     ofn.lpTemplateName         = NULL ;

     return 0 ;

case WM_SIZE:
     cxClient = LOWORD (lParam) ;
     cyClient = HIWORD (lParam) ;
     return 0 ;

case WM_COMMAND:
     switch (LOWORD (wParam))
     {
     case IDM_FILE_OPEN:

          // Show the File Open dialog box

          if (!GetOpenFileName (&ofn))
               return 0 ;
```

```
                // If there's an existing packed DIB, free the memory

            if (pPackedDib)
            {
                free (pPackedDib) ;
                pPackedDib = NULL ;
            }

                // If there's an existing logical palette, delete it

            if (hPalette)
            {
                DeleteObject (hPalette) ;
                hPalette = NULL ;
            }
                // Load the packed DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pPackedDib = PackedDibLoad (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (pPackedDib)
            {
                        // Create the palette from the DIB color table

                hPalette = PackedDibCreatePalette (pPackedDib) ;
            }
            else
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hPalette)
        {
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
```

```
        }

        if (pPackedDib)
            SetDIBitsToDevice (hdc,
                               0,
                               0,
                               PackedDibGetWidth (pPackedDib),
                               PackedDibGetHeight (pPackedDib),
                               0,
                               0,
                               0,
                               PackedDibGetHeight (pPackedDib),
                               PackedDibGetBitsPtr (pPackedDib),
                               pPackedDib,
                               DIB_RGB_COLORS) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_QUERYNEWPALETTE:
        if (!hPalette)
            return FALSE ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        InvalidateRect (hwnd, NULL, TRUE) ;

        ReleaseDC (hwnd, hdc) ;
        return TRUE ;

    case WM_PALETTECHANGED:
        if (!hPalette || (HWND) wParam == hwnd)
            break ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        UpdateColors (hdc) ;

        ReleaseDC (hwnd, hdc) ;
        break ;

    case WM_DESTROY:
        if (pPackedDib)
            free (pPackedDib) ;

        if (hPalette)
            DeleteObject (hPalette) ;
```

```
            PostQuitMessage (0) ;
            return 0 ;
        }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB3.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

SHOWDIB3 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",                       IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib3.rc

# define IDM_FILE_OPEN              40001
```

SHOWDIB3 中的窗口过程将紧缩 DIB 指针作为静态变量来维护，窗口过程在 File Open 命令期间调用 PACKEDIB.C 中的 PackedDibLoad 函数时获得此指针。在处理此命令的过程中，SHOWDIB3 也调用 PackedDibCreatePalette 来获得可能用于 DIB 的调色板。注意，无论 SHOWDIB3 什么时候准备加载新的 DIB，都应先释放前一个 DIB 的内存，并删除前一个 DIB 的调色板。在处理 WM_DESTROY 消息的过程中，最后的 DIB 最后释放，最后的调色板最后删除。

处理 WM_PAINT 消息很简单：如果存在调色板，则 SHOWDIB3 将它选进设备描述表并实现它。然后它调用 SetDIBitsToDevice，并传递有关 DIB 的函数信息（例如宽、高和指向 DIB 像素位的指针），这些信息从 PACKEDIB 中的函数获得。

另外，请记住 SHOWDIB3 基于 DIB 中的颜色表创建了调色板。如果在 DIB 中没有颜色表——通常是 16 位、24 位和 32 位 DIB 的情况——就不创建调色板。在 8 位视频模式下显示 DIB 时，它只能用标准保留的 20 种颜色显示。

对这个问题有两种解决方法：第一种是简单地使用“通用”调色板，这种调色板能适

用于许多图形。您也可以自己创建调色板;第二种解决方法是进入 DIB 的像素位,并决定要显示图像的最优颜色。很明显,第二种方法将涉及更多的工作(对于程序员和处理器都是如此),但是我将在本章结束之前告诉您如何使用第二种方法。

16.3.2 通用调色板

程序 16-14 所示的 SHOWDIB4 程序创建了一个通用的调色板,它用于显示加载到程序中的所有 DIB。另外,SHOWDIB4 与 SHOWDIB3 非常相似。

程序 16-14 SHOWDIB4 程序

SHOWDIB4.C

```
/* ------------------------------
    SHOWDIB4.C——Displays DIB with "all-purpose" palette
                (c) Charles Petzold, 1998
   ------------------------------ */

# include <windows.h>
# include "..\\ShowDib3\\PackeDib.h"
# include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib4") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    HWND        hwnd ;
    MSG         msg ;
    WNDCLASS    wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = szAppName ;
    wndclass.lpszClassName  = szAppName ;
    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
```

```
                    szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #4: All-Purpose Palette"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

/*-------------------------------------
     CreateAllPurposePalette: Creates a palette suitable for a wide variety
          of images; the palette has 247 entries, but 15 of them are
          duplicates or match the standard 20 colors.
   -------------------------------------*/

HPALETTE CreateAllPurposePalette (void)
{
     HPALETTE hPalette ;
     int              i, incr, R, G, B ;
     LOGPALETTE * plp ;

     plp = malloc (sizeof (LOGPALETTE) + 246 * sizeof (PALETTEENTRY)) ;

     plp -> palVersion    = 0x0300 ;
     plp -> palNumEntries = 247 ;

          // The following loop calculates 31 gray shades, but 3 of them
          //      will match the standard 20 colors
     for (i = 0, G = 0, incr = 8 ; G <= 0xFF ; i++, G += incr)
     {
          plp -> palPalEntry[i].peRed   = (BYTE) G ;
          plp -> palPalEntry[i].peGreen = (BYTE) G ;
          plp -> palPalEntry[i].peBlue  = (BYTE) G ;
          plp -> palPalEntry[i].peFlags = 0 ;

          incr = (incr == 9 ? 8 : 9) ;
```

```
    }

        // The following loop is responsible for 216 entries, but 8 of
        //       them will match the standard 20 colors, and another
        //       4 of them will match the gray shades above.
    for (R = 0 ; R <= 0xFF ; R += 0x33)
    for (G = 0 ; G <= 0xFF ; G += 0x33)
    for (B = 0 ; B <= 0xFF ; B += 0x33)
    {
        plp -> palPalEntry[i].peRed   = (BYTE) R ;
        plp -> palPalEntry[i].peGreen = (BYTE) G ;
        plp -> palPalEntry[i].peBlue  = (BYTE) B ;
        plp -> palPalEntry[i].peFlags = 0 ;

        i++ ;
    }
    hPalette = CreatePalette (plp) ;

    free (plp) ;
    return hPalette ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BITMAPINFO * pPackedDib ;
    static HPALETTE     hPalette ;
    static int          cxClient, cyClient ;
    static OPENFILENAME ofn ;
    static TCHAR        szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
    static TCHAR        szFilter[] = TEXT ("Bitmap Files (*.BMP)\0*.bmp\0")
                                     TEXT ("All Files (*.*)\0*.*\0\0") ;
    HDC                 hdc ;
    PAINTSTRUCT         ps ;
    switch (message)
    {
    case WM_CREATE:
        ofn.lStructSize       = sizeof (OPENFILENAME) ;
        ofn.hwndOwner         = hwnd ;
        ofn.hInstance         = NULL ;
        ofn.lpstrFilter       = szFilter ;
        ofn.lpstrCustomFilter = NULL ;
        ofn.nMaxCustFilter    = 0 ;
        ofn.nFilterIndex      = 0 ;
        ofn.lpstrFile         = szFileName ;
        ofn.nMaxFile          = MAX_PATH ;
        ofn.lpstrFileTitle    = szTitleName ;
        ofn.nMaxFileTitle     = MAX_PATH ;
        ofn.lpstrInitialDir   = NULL ;
```

```
        ofn.lpstrTitle          = NULL ;
        ofn.Flags               = 0 ;
        ofn.nFileOffset         = 0 ;
        ofn.nFileExtension      = 0 ;
        ofn.lpstrDefExt         = TEXT ("bmp") ;
        ofn.lCustData           = 0 ;
        ofn.lpfnHook            = NULL ;
        ofn.lpTemplateName      = NULL ;

            // Create the All-Purpose Palette

        hPalette = CreateAllPurposePalette () ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing packed DIB, free the memory

            if (pPackedDib)
            {
                free (pPackedDib) ;
                pPackedDib = NULL ;
            }

                // Load the packed DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pPackedDib = PackedDibLoad (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (!pPackedDib)
```

```
            }
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (pPackedDib)
        {
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;

            SetDIBitsToDevice (hdc,
                               0,
                               0,
                               PackedDibGetWidth (pPackedDib),
                               PackedDibGetHeight (pPackedDib),
                               0,
                               0,
                               0,
                               PackedDibGetHeight (pPackedDib),
                               PackedDibGetBitsPtr (pPackedDib),
                               pPackedDib,
                               DIB_RGB_COLORS) ;
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_QUERYNEWPALETTE:
        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        InvalidateRect (hwnd, NULL, TRUE) ;

        ReleaseDC (hwnd, hdc) ;
        return TRUE ;

    case WM_PALETTECHANGED:
        if ((HWND) wParam != hwnd)

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
```

```
            UpdateColors (hdc) ;

            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
            if (pPackedDib)
                free (pPackedDib) ;

            DeleteObject (hPalette) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB4.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

SHOWDIB4 MENU DISCARDABLE
BEGIN
    POPUP "&Open"
    BEGIN
        MENUITEM "&File",                       IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib4.rc

# define IDM_FILE_OPEN                  40001
```

在处理 WM_CREATE 消息时，SHOWDIB4 将调用 CreateAllPurposePalette，并在程序中保留该调色板，而在 WM_DESTROY 消息期间删除它。因为程序知道调色板总是存在的，所以在处理 WM_PAINT、WM_QUERYNEWPALETTE 或 WM_PALETTECHANGED 消息中，不必检查调色板的存在。

CreateAllPurposePalette 函数似乎创建具有 247 个条目的逻辑调色板，它超出了系统调色板中允许程序正常访问的 236 个条目。的确，此函数可以，而且还很方便。这些条

目中有 15 个被复制或者映射到 20 种标准的保留颜色中。

CreateAllPurposePalette 从创建 31 种灰色底纹开始，即红色、绿色和蓝色值为 0x00、0x09、0x11、0x1A、0x22、0x2B、0x33、0x3C、0x44、0x4D、0x55、0x5E、0x66、0x6F、0x77、0x80、0x88、0x91、0x99、0xA2、0xAA、0xB3、0xBB、0xC4、0xCC、0xD5、0xDD、0xE6、0xEE、0xF9，以及 0xFF。注意，第一个、最后一个和中间的条目都在标准的 20 种保留颜色中。下一个函数用红色、绿色和蓝色值的所有组合创建了颜色 0x00、0x33、0x66、0x99、0xCC 和 0xFF。这样就共有 216 种颜色，但是其中 8 种颜色复制了标准的 20 种保留颜色，而另外 4 种复制了前面计算的灰色底纹。如果将 PALETTEENTRY 结构的 peFlags 字段置为 0，则 Windows 将不把复制的条目放进系统调色板。

显然，实际的程序不希望计算 16 位、24 位或者 32 位 DIB 的最优调色板，程序将继续使用 DIB 颜色表来显示 8 位 DIB。SHOWDIB4 不完成这项工作，它只对每件事都使用通用调色板。因为 SHOWDIB4 是一个演示程序，而且您可以与 SHOWDIB3 显示的 8 位 DIB 进行比较。如果看一些人的彩色 DIB，那么您可能会得出这样的结论：SHOWDIB4 没有足够的颜色来精确地表示鲜艳的色调。

如果用 SHOWDIB4 中的 CreateAllPurposePalette 函数来试验（可能是通过将逻辑调色板的大小减少到只有几个条目的方法），您将发现当调色板选进设备描述表时，Windows 将只使用调色板中的颜色，而不使用标准的 20 种颜色调色板的颜色。

16.3.3 半色调调色板

Windows API 包括一个通用调色板（原文如此，但译者认为应为“半色调调色板”），程序可以通过调用 CreateHalftonePalette 来获得该调色板。使用此调色板的方法与使用从 SHOWDIB4 中的 CreateAllPurposePalette 获得调色板的方法相同，或者您也可以与位图伸展模式（称为 HALFTONE）——用 SetStretchBltMode 设置——一起使用。程序 16-15 所示的 SHOWDIB5 程序演示了使用半色调调色板的方法。

程序 16-15 SHOWDIB5 程序

SHOWDIB5.C

```
/*------------------------
   SHOWDIB5.C——Displays DIB with halftone palette
               (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"
```

```
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[ ] = TEXT ("ShowDib5") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #5: Halftone Palette"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
```

```
static BITMAPINFO * pPackedDib ;
static HPALETTE     hPalette ;
static int          cxClient, cyClient ;
static OPENFILENAME ofn ;
static TCHAR        szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
static TCHAR        szFilter[] = TEXT ("Bitmap Files (*.BMP)\0*.bmp\0")
                                 TEXT ("All Files (*.*)\0*.*\0\0") ;
HDC                 hdc ;
PAINTSTRUCT         ps ;

switch (message)
{
case WM_CREATE:
     ofn.lStructSize       = sizeof (OPENFILENAME) ;
     ofn.hwndOwner         = hwnd ;
     ofn.hInstance         = NULL ;
     ofn.lpstrFilter       = szFilter ;
     ofn.lpstrCustomFilter = NULL ;
     ofn.nMaxCustFilter    = 0 ;
     ofn.nFilterIndex      = 0 ;
     ofn.lpstrFile         = szFileName ;
     ofn.nMaxFile          = MAX_PATH ;
     ofn.lpstrFileTitle    = szTitleName ;
     ofn.nMaxFileTitle     = MAX_PATH ;
     ofn.lpstrInitialDir   = NULL ;
     ofn.lpstrTitle        = NULL ;
     ofn.Flags             = 0 ;
     ofn.nFileOffset       = 0 ;
     ofn.nFileExtension    = 0 ;
     ofn.lpstrDefExt       = TEXT ("bmp") ;
     ofn.lCustData         = 0 ;
     ofn.lpfnHook          = NULL ;
     ofn.lpTemplateName    = NULL ;

          // Create the All-Purpose Palette

     hdc = GetDC (hwnd) ;
     hPalette = CreateHalftonePalette (hdc) ;
     ReleaseDC (hwnd, hdc) ;
     return 0 ;

case WM_SIZE:
     cxClient = LOWORD (lParam) ;
     cyClient = HIWORD (lParam) ;
     return 0 ;

case WM_COMMAND:
```

```
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing packed DIB, free the memory
            if (pPackedDib)
            {
                free (pPackedDib) ;
                pPackedDib = NULL ;
            }

                // Load the packed DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pPackedDib = PackedDibLoad (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (!pPackedDib)
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (pPackedDib)
        {
                // Set halftone stretch mode

            SetStretchBltMode (hdc, HALFTONE) ;
            SetBrushOrgEx (hdc, 0, 0, NULL) ;

                // Select and realize halftone palette
```

```
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;

                // StretchDIBits rather than SetDIBitsToDevice
            StretchDIBits (hdc,
                           0,
                           0,
                           PackedDibGetWidth (pPackedDib),
                           PackedDibGetHeight (pPackedDib),
                           0,
                           0,
                           PackedDibGetWidth (pPackedDib),
                           PackedDibGetHeight (pPackedDib),
                           PackedDibGetBitsPtr (pPackedDib),
                           pPackedDib,
                           DIB_RGB_COLORS,
                           SRCCOPY) ;
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_QUERYNEWPALETTE:
        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        InvalidateRect (hwnd, NULL, TRUE) ;

        ReleaseDC (hwnd, hdc) ;
        return TRUE ;

    case WM_PALETTECHANGED:
        if ((HWND) wParam != hwnd)

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        UpdateColors (hdc) ;

        ReleaseDC (hwnd, hdc) ;
        break ;

    case WM_DESTROY:
        if (pPackedDib)
            free (pPackedDib) ;

        DeleteObject (hPalette) ;

        PostQuitMessage (0) ;
        return 0 ;
```

```
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB5.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

SHOWDIB5 MENU DISCARDABLE
BEGIN
    POPUP "&Open"
    BEGIN
        MENUITEM "&File",                       IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib5.rc

#define IDM_FILE_OPEN               40001
```

SHOWDIB5 程序类似于 SHOWDIB4，SHOWDIB4 中不使用 DIB 中的颜色表，而使用适合于图像范围更大的调色板。为此，SHOWDIB5 使用了由 Windows 支持的逻辑调色板，其句柄可以从 CreateHalftonePalette 函数获得。

半色调调色板并不比 SHOWDIB4 中的 CreateAllPurposePalette 函数所创建的调色板更复杂。的确，如果它自己使用，结果是相同的。然而，如果您调用下面两个函数：

```
SetStretchBltMode (hdc, HALFTONE) ;
SetBrushOrgEx (hdc, x, y, NULL) ;
```

其中，x 和 y 是 DIB 左上角的设备坐标，并且如果您用 StretchDIBits 而不是 SetDIBitsToDevice 来显示 DIB，那么结果会让您吃惊：颜色刷新色调要比不设置位图伸展模式来使用 CreateAllPurposePalette 或者 CreateHalftonePalette 更精确。Windows 使用一种混色图案来处理半色调调色板上的颜色，以使其更接近 8 位显示卡上原始图像的颜色。与您想象的一样，这样做的缺点是要用更多的处理时间。

16.3.4　索引调色板颜色

现在来讨论 SetDIBitsToDevice、StretchDIBits、CreateDIBitmap、SetDIBits、GetDIBits，以及 Cre-

ateDIBSection 的 fClrUse 参数。通常，您将这个参数设置为 DIB_RGB_COLORS(等于 0)。不过，您也能将它设置为 DIB_PAL_COLORS。在这种情况下，假定 BITMAPINFO 结构中的颜色表不包括 RGB 颜色值，而是在逻辑调色板中包括 16 位索引。逻辑调色板是当前在设备描述表中选择的，并作为第一个参数传递给函数。实际上，在 CreateDIBSection 中，之所以需要指定一个非 NULL 的设备描述表句柄作为第一个参数，只是因为使用 DIB_PAL_COLORS。

DIB_PAL_COLORS 能为您做些什么呢？它可能提高一些性能。考虑一下在 8 位视频模式下调用 SetDIBitsToDevice 显示的 8 位 DIB。Windows 必须首先在 DIB 颜色表的所有颜色中搜索与设备可用颜色最接近的颜色，然后设置一个小表，以便将 DIB 像素值映射到设备像素。也就是说，最多需要搜索 256 次最接近的颜色。但是，如果 DIB 颜色表中含有从设备描述表中选择的逻辑调色板中的替代索引，那么就可能跳过搜索。

除了使用调色板索引以外，程序 16-16 所示的 SHOWDIB6 程序与 SHOWDIB3 相似。

程序 16-16　SHOWDIB6 程序

SHOWDIB6.C

```
/*-----------------------
    SHOWDIB6.C——Display DIB with palette indices
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib6") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
```

```
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #6: Palette Indices"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BITMAPINFO * pPackedDib ;
     static HPALETTE     hPalette ;
     static int          cxClient, cyClient ;
     static OPENFILENAME ofn ;
     static TCHAR        szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR        szFilter[] = TEXT ("Bitmap Files (*.BMP) \0*.bmp \0")
                                      TEXT ("All Files (*.*) \0*.* \0\0") ;
     HDC                 hdc ;
     int                 i, iNumColors ;
     PAINTSTRUCT         ps ;
     WORD          *     pwIndex ;

     switch (message)
     {
     case WM_CREATE:
          ofn.lStructSize          = sizeof (OPENFILENAME) ;
```

```
        ofn.hwndOwner         = hwnd ;
        ofn.hInstance         = NULL ;
        ofn.lpstrFilter       = szFilter ;
        ofn.lpstrCustomFilter = NULL ;
        ofn.nMaxCustFilter    = 0 ;
        ofn.nFilterIndex      = 0 ;
        ofn.lpstrFile         = szFileName ;
        ofn.nMaxFile          = MAX_PATH ;
        ofn.lpstrFileTitle    = szTitleName ;
        ofn.nMaxFileTitle     = MAX_PATH ;
        ofn.lpstrInitialDir   = NULL ;
        ofn.lpstrTitle        = NULL ;
        ofn.Flags             = 0 ;
        ofn.nFileOffset       = 0 ;
        ofn.nFileExtension    = 0 ;
        ofn.lpstrDefExt       = TEXT ("bmp") ;
        ofn.lCustData         = 0 ;
        ofn.lpfnHook          = NULL ;
        ofn.lpTemplateName    = NULL ;

        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box
            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing packed DIB, free the memory

            if (pPackedDib)
            {
                free (pPackedDib) ;
                pPackedDib = NULL ;
            }

                // If there's an existing logical palette, delete it

            if (hPalette)
            {
```

```
                DeleteObject (hPalette) ;
                hPalette = NULL ;
            }

                // Load the packed DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pPackedDib = PackedDibLoad (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (pPackedDib)
            {
                    // Create the palette from the DIB color table

                hPalette = PackedDibCreatePalette (pPackedDib) ;

                    // Replace DIB color table with indices

                if (hPalette)
                {
                    iNumColors = PackedDibGetNumColors (pPackedDib) ;
                    pwIndex = (WORD *)
                                PackedDibGetColorTablePtr (pPackedDib) ;

                    for (i = 0 ; i < iNumColors ; i++)
                        pwIndex[i] = (WORD) i ;
                }
            }
            else
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hPalette)
        {
            SelectPalette (hdc, hPalette, FALSE) ;
```

```
            RealizePalette (hdc) ;
        }

        if (pPackedDib)
            SetDIBitsToDevice (hdc,
                               0,
                               0,
                               PackedDibGetWidth (pPackedDib),
                               PackedDibGetHeight (pPackedDib),
                               0,
                               0,
                               0,
                               PackedDibGetHeight (pPackedDib),
                               PackedDibGetBitsPtr (pPackedDib),
                               pPackedDib,
                               DIB_PAL_COLORS) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_QUERYNEWPALETTE:
        if (!hPalette)
            return FALSE ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        InvalidateRect (hwnd, NULL, TRUE) ;

        ReleaseDC (hwnd, hdc) ;
        return TRUE ;

    case WM_PALETTECHANGED:
        if (!hPalette || (HWND) wParam == hwnd)
            break ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        UpdateColors (hdc) ;

        ReleaseDC (hwnd, hdc) ;
        break ;

    case WM_DESTROY:
        if (pPackedDib)
            free (pPackedDib) ;

        if (hPalette)
```

```
            DeleteObject (hPalette) ;

        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB6.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

SHOWDIB6 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",                       IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib6.rc
//
#define IDM_FILE_OPEN                   40001
```

SHOWDIB6 将 DIB 加载到内存并由此创建了调色板以后，SHOWDIB6 简单地用以 0 开始的 WORD 索引替换 DIB 颜色表中的颜色。PackedDibGetNumColors 函数将表示有多少种颜色，而 PackedDibGetColorTablePtr 函数返回指向 DIB 颜色表起始位置的指针。

注意，只有直接从 DIB 颜色表来创建调色板时，此技术才可行。如果使用通用调色板，则必须搜索最接近的颜色，以获得放入 DIB 的索引。

如果要使用调色板索引，那么请在将 DIB 保存到磁盘之前，确保替换了 DIB 中的颜色表。另外，不要将包含调色板索引的 DIB 放入剪贴板。实际上，在显示之前，将调色板索引放入 DIB，然后将 RGB 颜色值放回，会更安全一些。

16.3.5 调色板和位图对象

程序 16-17 中的 SHOWDIB7 程序显示了如何使用与 DIB 相关联的调色板，这些 DIB 是使用 CreateDIBitmap 函数转换成 GDI 位图对象的。

程序 16-17 SHOWDIB7 程序

SHOWDIB7.C

```
/*-----------------------
   SHOWDIB7.C——Shows DIB converted to DDB
                (c) Charles Petzold, 1998
   -----------------------*/
#include <windows.h>
#include "..\\ShowDib3\\PackeDib.h"
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib7") ;
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Show DIB #7: Converted to DDB"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
```

```
    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HBITMAP        hBitmap ;
    static HPALETTE       hPalette ;
    static int            cxClient, cyClient ;
    static OPENFILENAME   ofn ;
    static TCHAR          szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
    static TCHAR          szFilter[] = TEXT ("Bitmap Files (*.BMP) \0*.bmp \0")
                                       TEXT ("All Files (*.*) \0*.* \0\0") ;
    BITMAP                bitmap ;
    BITMAPINFO          * pPackedDib ;
    HDC                   hdc, hdcMem ;
    PAINTSTRUCT           ps ;

    switch (message)
    {
    case WM_CREATE:
        ofn.lStructSize       = sizeof (OPENFILENAME) ;
        ofn.hwndOwner         = hwnd ;
        ofn.hInstance         = NULL ;
        ofn.lpstrFilter       = szFilter ;
        ofn.lpstrCustomFilter = NULL ;
        ofn.nMaxCustFilter    = 0 ;
        ofn.nFilterIndex      = 0 ;
        ofn.lpstrFile         = szFileName ;
        ofn.nMaxFile          = MAX_PATH ;
        ofn.lpstrFileTitle    = szTitleName ;
        ofn.nMaxFileTitle     = MAX_PATH ;
        ofn.lpstrInitialDir   = NULL ;
        ofn.lpstrTitle        = NULL ;
        ofn.Flags             = 0 ;
        ofn.nFileOffset       = 0 ;
        ofn.nFileExtension    = 0 ;
        ofn.lpstrDefExt       = TEXT ("bmp") ;
        ofn.lCustData         = 0 ;
        ofn.lpfnHook          = NULL ;
        ofn.lpTemplateName    = NULL ;
```

```
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing packed DIB, free the memory
            if (hBitmap)
            {
                DeleteObject (hBitmap) ;
                hBitmap = NULL ;
            }

                // If there's an existing logical palette, delete it

            if (hPalette)
            {
                DeleteObject (hPalette) ;
                hPalette = NULL ;
            }

                // Load the packed DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pPackedDib = PackedDibLoad (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (pPackedDib)
            {
                    // Create palette from the DIB and select it into DC

                hPalette = PackedDibCreatePalette (pPackedDib) ;
```

```
                hdc = GetDC (hwnd) ;

                if (hPalette)
                {
                     SelectPalette (hdc, hPalette, FALSE) ;
                     RealizePalette (hdc) ;
                }
                    // Create the DDB from the DIB
                hBitmap = CreateDIBitmap (hdc,
                                          (PBITMAPINFOHEADER) pPackedDib,
                                          CBM_INIT,
                                          PackedDibGetBitsPtr (pPackedDib),
                                          pPackedDib,
                                          DIB_RGB_COLORS) ;
                ReleaseDC (hwnd, hdc) ;

                    // Free the packed-DIB memory

                free (pPackedDib) ;
           }
           else
           {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
           }
           InvalidateRect (hwnd, NULL, TRUE) ;
           return 0 ;
      }
      break ;

case WM_PAINT:
      hdc = BeginPaint (hwnd, &ps) ;

      if (hPalette)
      {
           SelectPalette (hdc, hPalette, FALSE) ;
           RealizePalette (hdc) ;
      }
      if (hBitmap)
      {
           GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;

           hdcMem = CreateCompatibleDC (hdc) ;
           SelectObject (hdcMem, hBitmap) ;

           BitBlt (hdc,      0, 0, bitmap.bmWidth, bitmap.bmHeight,
                hdcMem, 0, 0, SRCCOPY) ;
```

```
                DeleteDC (hdcMem) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_QUERYNEWPALETTE:
            if (!hPalette)
                return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

        case WM_PALETTECHANGED:
            if (!hPalette || (HWND) wParam == hwnd)
                break ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            UpdateColors (hdc) ;

            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
            if (hBitmap)
                DeleteObject (hBitmap) ;

            if (hPalette)
                DeleteObject (hPalette) ;

            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB7.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"
```

```
//////////////////////////////////////////////////////////////////////////////
// Menu

SHOWDIB7 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",                       IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib7.rc

#define IDM_FILE_OPEN               40001
```

与前面的程序一样,SHOWDIB7 获得了一个指向紧缩 DIB 的指针,该 DIB 响应菜单中的 File Open 命令。程序还通过紧缩 DIB 创建一个调色板,然后——还是在 WM_COMMAND 消息的处理过程中——获得用于视频显示器的设备描述表,并选进调色板,实现调色板。然后,SHOWDIB7 调用 CreateDIBitmap,以便从 DIB 创建 DDB。如果调色板没有选进并实现设备描述表,那么 CreateDIBitmap 创建的 DDB 将不使用逻辑调色板中的其他颜色。

调用 CreateDIBitmap 以后,该程序将释放紧缩 DIB 占用的内存空间。pPackedDib 变量不是静态变量。相反,SHOWDIB7 用静态变量保留了位图句柄(hBitmap)和逻辑调色板句柄(hPalette)。

在处理 WM_PAINT 消息期间,调色板再次被选进设备描述表并实现。从 GetObject 函数可以获得位图的宽度和高度,然后,程序会通过创建一个兼容的内存设备描述表在客户区中显示位图,并将该位图选进所创建的内存设备表,然后执行 BitBlt。显示 DDB 时所用的调色板,必须与从 CreateDIBitmap 调用创建时所用的调色板相同。

如果将位图复制到剪贴板,则最好使用紧缩 DIB 格式,然后 Windows 可以将位图对象提供给希望使用这些位图的程序。然而,如果需要将位图对象复制到剪贴板,则首先要获得视频设备描述表并实现调色板。这允许 Windows 基于当前的系统调色板将 DDB 转换为 DIB。

16.3.6　调色板和 DIB 块

最后,程序 16-18 所示的 SHOWDIB8 显示了如何使用带有 DIB 块的调色板。

程序 16-18 SHOWDIB8 程序

SHOWDIB8.C

```
/* ---------------------------
    SHOWDIB8.C——Shows DIB converted to DIB section
                (c) Charles Petzold, 1998
   ---------------------------*/

# include <windows.h>
# include "..\\ ShowDib3 \\ PackeDib.h"
# include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("ShowDib8") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    HWND        hwnd ;
    MSG         msg ;
    WNDCLASS    wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = szAppName ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Show DIB #8: DIB Section"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HBITMAP        hBitmap ;
     static HPALETTE       hPalette ;
     static int            cxClient, cyClient ;
     static OPENFILENAME   ofn ;
     static PBYTE          pBits ;
     static TCHAR          szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     static TCHAR          szFilter[] = TEXT ("Bitmap Files (*.BMP) \0*.bmp \0")
                                        TEXT ("All Files (*.*) \0*.* \0\0") ;
     BITMAP                bitmap ;
     BITMAPINFO            *pPackedDib ;
     HDC                   hdc, hdcMem ;
     PAINTSTRUCT           ps ;

     switch (message)
     {
     case WM_CREATE:
          ofn.lStructSize       = sizeof (OPENFILENAME) ;
          ofn.hwndOwner         = hwnd ;
          ofn.hInstance         = NULL ;
          ofn.lpstrFilter       = szFilter ;
          ofn.lpstrCustomFilter = NULL ;
          ofn.nMaxCustFilter    = 0 ;
          ofn.nFilterIndex      = 0 ;
          ofn.lpstrFile         = szFileName ;
          ofn.nMaxFile          = MAX_PATH ;
          ofn.lpstrFileTitle    = szTitleName ;
          ofn.nMaxFileTitle     = MAX_PATH ;
          ofn.lpstrInitialDir   = NULL ;
          ofn.lpstrTitle        = NULL ;
          ofn.Flags             = 0 ;
          ofn.nFileOffset       = 0 ;
          ofn.nFileExtension    = 0 ;
          ofn.lpstrDefExt       = TEXT ("bmp") ;
```

```
        ofn.lCustData           = 0 ;
        ofn.lpfnHook            = NULL ;
        ofn.lpTemplateName      = NULL ;

        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing packed DIB, free the memory

            if (hBitmap)
            {
                DeleteObject (hBitmap) ;
                hBitmap = NULL ;
            }

                // If there's an existing logical palette, delete it

            if (hPalette)
            {
                DeleteObject (hPalette) ;
                hPalette = NULL ;
            }
                // Load the packed DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            pPackedDib = PackedDibLoad (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (pPackedDib)
            {
```

```
                    // Create the DIB section from the DIB

                hBitmap = CreateDIBSection (NULL,
                                            pPackedDib,
                                            DIB_RGB_COLORS,
                                            &pBits,
                                            NULL, 0) ;

                    // Copy the bits

                CopyMemory (pBits, PackedDibGetBitsPtr (pPackedDib),
                                   PackedDibGetBitsSize (pPackedDib)) ;

                    // Create palette from the DIB

                hPalette = PackedDibCreatePalette (pPackedDib) ;

                    // Free the packed-DIB memory

                free (pPackedDib) ;
            }
            else
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file"),
                            szAppName, 0) ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        break ;
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hPalette)
        {
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
        }
        if (hBitmap)
        {
            GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;

            hdcMem = CreateCompatibleDC (hdc) ;
            SelectObject (hdcMem, hBitmap) ;

            BitBlt (hdc,      0, 0, bitmap.bmWidth, bitmap.bmHeight,
                    hdcMem, 0, 0, SRCCOPY) ;
            DeleteDC (hdcMem) ;
```

```
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_QUERYNEWPALETTE:
            if (!hPalette)
                return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

        case WM_PALETTECHANGED:
            if (!hPalette || (HWND) wParam == hwnd)
                break ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            UpdateColors (hdc) ;

            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
            if (hBitmap)
                DeleteObject (hBitmap) ;

            if (hPalette)
                DeleteObject (hPalette) ;

            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

SHOWDIB8.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
```

```
// Menu

SHOWDIB8 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open",                    IDM_FILE_OPEN
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by ShowDib8.rc

# define IDM_FILE_OPEN              40001
```

在 SHOWDIB7 和 SHOWDIB8 中的 WM_PAINT 处理是一样的:两个程序都将位图句柄(hBitmap)和逻辑调色板句柄(hPalette)作为静态变量。调色板被选进设备描述表并实现,位图的宽度和高度从 GetObject 函数获得,程序创建内存设备描述表并选进位图,然后通过调用 BitBlt 将位图显示到客户区。

两个程序之间最大的差别在于处理 File、Open 菜单命令的过程。在获得指向紧缩 DIB 的指针并创建了调色板以后,SHOWDIB7 必须将调色板选进视频设备描述表,并在调用 CreateDIBitmap 之前实现。SHOWDIB8 在获得紧缩 DIB 指针以后调用 CreateDIBSection。不必将调色板选进设备描述表,这是因为 CreateDIBSection 不将 DIB 转换成设备相关的格式。的确,CreateDIBSection(即设备描述表句柄)的第一个参数的唯一用途在于您是否使用 DIB_PAL_COLORS 标志。

调用 CreateDIBSection 以后,SHOWDIB8 将像素位从紧缩 DIB 复制到从 CreateDIBSection 函数返回的内存位置,然后调用 PackedDibCreatePalette。尽管此函数对于程序来说使用非常方便,但是 SHOWDIB8 可以已经根据从 GetDIBColorTable 函数返回的信息创建了一个调色板。

16.4 用于 DIB 的库

只是现在——经过我们长时间地学习 GDI 位图对象、与设备无关的位图、DIB 块,以及 Windows 调色板管理器之后——我们才做好了开发一套有助于处理位图的函数的准备。

前面的 PACKEDIB 文件演示了一种可能的方法:内存中的紧缩 DIB 只用指向它的指针表示。程序需要的有关 DIB 的全部信息都可以从访问头信息结构的函数获得。然而,实际上到"获取像素"和"设置像素"例程时,这种方法就会产生严重的性能问题。图

像处理任务当然需要访问位图的字位,并且这些函数也应该尽可能得快。

可能 C++ 的解决方法中包括创建 DIB 类,这时指向紧缩 DIB 的指针正好是一个成员变量。其他成员变量和成员函数有助于更快地执行获得和设置 DIB 中的像素的例程。不过,因为在第一章已经指出,对于本书您只需要了解 C,使用 C++ 将是其他书的解决方法。

当然,用 C++ 能做的事情用 C 也能做。一个好的例子就是许多 Windows 函数都使用句柄。除了将句柄当作数值以外,应用程序对它还了解什么呢? 程序知道句柄引用特殊的函数对象,还知道函数用于处理现存的对象。显然,操作系统按某种方式用句柄来引用对象的内部信息。句柄可以与结构指针一样简单。

例如,假设有一个函数集,这些函数都使用一个被称为 HDIB 的句柄。HDIB 是什么呢? 它可能在某个头文件中定义如下:

```
typedef void * HDIB ;
```

此定义用"不关您的事"回答了"HDIB 是什么"这个问题。

然而,实际上 HDIB 可能是结构指针,该结构不仅包括指向紧缩 DIB 的指针,还包括其他信息:

```
typedef struct
{
     BITMAPINFO * pPackedDib ;
     int          cx, cy, cBitsPerPixel, cBytesPerRow ;
     BYTE       * pBits ;
}
DIBSTRUCTURE, * PDIBSTRUCTURE ;
```

此结构的其他 5 个字段包括从紧缩 DIB 中引出的信息,当然,结构中这些值允许更快地访问它们。不同的 DIB 库函数都可以处理这个结构,而不是 pPackedDib 指针。可以按下面的方法来执行 DibGetPixelPointer 函数:

```
BYTE * DibGetPixelPointer (HDIB hdib, int x, int y)
{
     PDIBSTRUCTURE pdib = hdib ;

     return pdib -> pBits + y * pdib -> cBytesPerRow +
                            x * pdib -> cBitsPerPixel / 8 ;
}
```

当然,这种方法可能要比 PACKEDIB.C 中执行"获取像素"例程快。

由于这种方法非常合理,所以我决定放弃紧缩 DIB,并代替基于 DIB 块的 DIB 库。这实际上使我们对紧缩 DIB 的处理有更大的弹性(也就是说,能够在设备无关的方式下操纵 DIB 像素位),而且在 Windows NT 下运行时将更有效。

16.4.1 DIBSTRUCT 结构

DIBHELP.C 文件——如此命名是因为对处理 DIB 提供帮助——有上千行，将分成几部分列在书中。但是让我们首先看一下 DIBHELP 函数所处理的结构，该结构在 DIBHELP.C 中定义如下：

```
typedef struct
{
    PBYTE       * ppRow ;       // array of row pointers
    int           iSignature ;  // = "Dib"
    HBITMAP       hBitmap ;     // handle returned from CreateDIBSection
    BYTE        * pBits ;       // pointer to bitmap bits
    DIBSECTION    ds ;          // DIBSECTION structure
    int           iRShift[3] ;  // right-shift values for color masks
    int           iLShift[3] ;  // left-shift values for color masks
}
DIBSTRUCT, * PDIBSTRUCT ;
```

现在跳过第一个字段。它之所以为第一个字段是因为它使某些宏更容易——在讨论完其他字段以后再来理解第一个字段就更容易了。

在 DIBHELP.C 中，当 DIB 创建的函数首先设置了此结构时，第二个字段就设置为文本字符串"Dib"的二进制值。这将被某些 DIBHELP 函数用作指向该结构的指针的有效性检查。

第三个字段，即 hBitmap，是从 CreateDIBSection 函数返回的位图句柄。您将想起该句柄可有多种使用方式，它与我们在第十四章遇到的 GDI 位图对象的句柄用法一样。不过，从 CreateDIBSection 返回的句柄将涉及按设备无关格式保存的位图，该位图格式一直保存到通过调用 BitBlt 和 StretchBlt 来将位图画到输出设备。

DIBSTRUCT 的第 4 个字段是指向位图位的指针。此值也可由 CreateDIBSection 函数设置。您将想起，操作系统将控制这个内存块，但应用程序有访问它的权限。在删除位图句柄时，块将自动释放。

DIBSTRUCT 的第 5 个字段是 DIBSECTION 结构。如果有从 CreateDIBSection 返回的位图句柄，那么可以将句柄传递给 GetObject 函数，以获得有关 DIBSECTION 结构中的位图信息：

```
GetObject (hBitmap, sizeof (DIBSECTION), &ds) ;
```

作为提示，DIBSECTION 结构在 WINGDI.H 中定义如下：

```
typedef struct tagDIBSECTION
{
    BITMAP              dsBm ;
    BITMAPINFOHEADER    dsBmih ;
```

```
    DWORD           dsBitfields[3] ;        // Color masks
    HANDLE          dshSection ;
    DWORD           dsOffset ;
}
DIBSECTION, * PDIBSECTION ;
```

第一个字段是 BITMAP 结构，它与 CreateBitmapIndirect 一起创建位图对象，与 GetObject 一起返回关于 DDB 的信息。第二个字段是 BITMAPINFOHEADER 结构。不管位图信息结构是否传递给 CreateDIBSection 函数，DIBSECTION 结构总有 BITMAPINFOHEADER 结构而不是其他结构，例如 BITMAPCOREHEADER 结构。这意味着在访问此结构时，DIBHELP.C 中的许多函数都不必检查与 OS/2 兼容的 DIB。

对于 16 位和 32 位的 DIB，如果 BITMAPINFOHEADER 结构的 biCompression 字段是 BI_BITFIELDS，那么在信息头结构后面通常有三个掩码值。这些掩码值决定如何将 16 位和 32 位像素值转换成 RGB 颜色。掩码保存在 DIBSECTION 结构的第三个字段中。

DIBSECTION 结构的最后两个字段指的是和文件映射目标一起创建的 DIB 块。DIBHELP 不使用 CreateDIBSection 的这个特征，因此可以忽略这些字段。

DIBSTRUCT 的最后两个字段保存左右移位值，这些值用于处理 16 位和 32 位 DIB 的颜色掩码。我们将在第十五章讨论这些移位值。

让我们再返回来看一下 DIBSTRUCT 的第一个字段。正如我们所看到的一样，在开始创建 DIB 时，此字段设置为指向一个指针数组的指针，该数组中的每个指针都指向 DIB 中的一行像素。这些指针允许以更快的方式来获得 DIB 像素位，同时也被定义，以便顶行可以首先引用 DIB 像素位。此数组的最后一个元素——引用 DIB 图像的最底行——通常等于 DIBSTRUCT 的 pBits 字段。

16.4.2 信息函数

DIBHELP.C 以定义 DIBSTRUCT 结构开始，然后提供一个函数集，此函数集允许应用程序获得有关 DIB 节的信息。程序 16-19 是 DIBHELP.C 的第一部分。

程序 16-19 DIBHELP.C 文件的第一部分

DIBHELP.C（第一部分）

```
/*-------------------------
   DIBHELP.C——DIB Section Helper Routines
              (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include "dibhelp.h"
```

```
# define HDIB_SIGNATURE ( * (int *) "Dib ")

typedef struct
{
 PBYTE       * ppRow ;          // must be first field for macros!
 int           iSignature ;
 HBITMAP       hBitmap ;
 BYTE        * pBits ;
 DIBSECTION    ds ;
 int           iRShift[3] ;
 int           iLShift[3] ;
}
DIBSTRUCT, * PDIBSTRUCT ;

/*-------------------------------
     DibIsValid: Returns TRUE if hdib points to a valid DIBSTRUCT
   -------------------------------*/

BOOL DibIsValid (HDIB hdib)
{
     PDIBSTRUCT pdib = hdib ;

     if (pdib == NULL)
         return FALSE ;

     if (IsBadReadPtr (pdib, sizeof (DIBSTRUCT)))
         return FALSE ;

     if (pdib -> iSignature != HDIB_SIGNATURE)
         return FALSE ;
     return TRUE ;
}

/*-------------------------------
     DibBitmapHandle: Returns the handle to the DIB section bitmap object
   -------------------------------*/

HBITMAP DibBitmapHandle (HDIB hdib)
{
     if (!DibIsValid (hdib))
         return NULL ;

     return ((PDIBSTRUCT) hdib) -> hBitmap ;
}

/*-------------------
     DibWidth: Returns the bitmap pixel width
   -------------------*/
```

```
int DibWidth (HDIB hdib)
{
     if (!DibIsValid (hdib))
          return 0 ;

     return ((PDIBSTRUCT) hdib) -> ds.dsBm.bmWidth ;
}

/* - - - - - - - - - - - - - - - - - - -
     DibHeight: Returns the bitmap pixel height
   - - - - - - - - - - - - - - - - - - - */

int DibHeight (HDIB hdib)
{
     if (!DibIsValid (hdib))
          return 0 ;

     return ((PDIBSTRUCT) hdib) -> ds.dsBm.bmHeight ;
}

/* - - - - - - - - - - - - - - - - - - - -
     DibBitCount: Returns the number of bits per pixel
   - - - - - - - - - - - - - - - - - - - - */

int DibBitCount (HDIB hdib)
{
     if (!DibIsValid (hdib))
          return 0 ;

     return ((PDIBSTRUCT) hdib) -> ds.dsBm.bmBitsPixel ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - -
     DibRowLength: Returns the number of bytes per row of pixels
   - - - - - - - - - - - - - - - - - - - - - - - - */

int DibRowLength (HDIB hdib)
{
     if (!DibIsValid (hdib))
          return 0 ;

     return 4 * ((DibWidth (hdib) * DibBitCount (hdib) + 31) / 32) ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - -
     DibNumColors: Returns the number of colors in the color table
   - - - - - - - - - - - - - - - - - - - - - - - - */

int DibNumColors (HDIB hdib)
```

```
{
    PDIBSTRUCT pdib = hdib ;

    if (!DibIsValid (hdib))
        return 0 ;

    if (pdib -> ds.dsBmih.biClrUsed != 0)
    {
        return pdib -> ds.dsBmih.biClrUsed ;
    }
    else if (DibBitCount (hdib) <= 8)
    {
        return 1 << DibBitCount (hdib) ;
    }
    return 0 ;
}

/*-----------------
    DibMask: Returns one of the color masks
   -----------------*/

DWORD DibMask (HDIB hdib, int i)
{
    PDIBSTRUCT pdib = hdib ;

    if (!DibIsValid (hdib) || i < 0 || i > 2)
        return 0 ;

    return pdib -> ds.dsBitfields[i] ;
}

/*-------------------
    DibRShift: Returns one of the right-shift values
   -------------------*/

int DibRShift (HDIB hdib, int i)
{
    PDIBSTRUCT pdib = hdib ;

    if (!DibIsValid (hdib) || i < 0 || i > 2)
        return 0 ;

    return pdib -> iRShift[i] ;
}

/*-------------------
    DibLShift: Returns one of the left-shift values
   -------------------*/
```

```
int DibLShift (HDIB hdib, int i)
{
     PDIBSTRUCT pdib = hdib ;

     if (!DibIsValid (hdib) || i < 0 || i > 2)
          return 0 ;

     return pdib -> iLShift[i] ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - -
     DibCompression: Returns the value of the biCompression field
   - - - - - - - - - - - - - - - - - - - - - - - - - */

int DibCompression (HDIB hdib)
{
     if (!DibIsValid (hdib))
          return 0 ;

     return ((PDIBSTRUCT) hdib) -> ds.dsBmih.biCompression ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - -
     DibIsAddressable: Returns TRUE if the DIB is not compressed
   - - - - - - - - - - - - - - - - - - - - - - - - - */

BOOL DibIsAddressable (HDIB hdib)
{
     int iCompression ;

     if (!DibIsValid (hdib))
          return FALSE ;

     iCompression = DibCompression (hdib) ;

     if (iCompression == BI_RGB || iCompression == BI_BITFIELDS)
          return TRUE ;

     return FALSE ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     These functions return the sizes of various components of the DIB section
       AS THEY WOULD APPEAR in a packed DIB. These functions aid in converting
       the DIB section to a packed DIB and in saving DIB files.
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

DWORD DibInfoHeaderSize (HDIB hdib)
```

```
{
    if (!DibIsValid (hdib))
        return 0 ;

    return ((PDIBSTRUCT) hdib) -> ds.dsBmih.biSize ;
}

DWORD DibMaskSize (HDIB hdib)
{
    PDIBSTRUCT pdib = hdib ;
    if (!DibIsValid (hdib))
        return 0 ;

    if (pdib -> ds.dsBmih.biCompression == BI_BITFIELDS)
        return 3 * sizeof (DWORD) ;

    return 0 ;
}

DWORD DibColorSize (HDIB hdib)
{
    return DibNumColors (hdib) * sizeof (RGBQUAD) ;
}

DWORD DibInfoSize (HDIB hdib)
{
    return DibInfoHeaderSize(hdib) + DibMaskSize(hdib) + DibColorSize(hdib) ;
}

DWORD DibBitsSize (HDIB hdib)
{
    PDIBSTRUCT pdib = hdib ;

    if (!DibIsValid (hdib))
        return 0 ;

    if (pdib -> ds.dsBmih.biSizeImage != 0)
    {
        return pdib -> ds.dsBmih.biSizeImage ;
    }
    return DibHeight (hdib) * DibRowLength (hdib) ;
}

DWORD DibTotalSize (HDIB hdib)
{
    return DibInfoSize (hdib) + DibBitsSize (hdib) ;
}
```

```
/* ---------------------------------------
    These functions return pointers to the various components of the DIB
      section.
   ---------------------------------------*/

BITMAPINFOHEADER * DibInfoHeaderPtr (HDIB hdib)
{
    if (!DibIsValid (hdib))
        return NULL ;

    return & (((PDIBSTRUCT) hdib) -> ds.dsBmih) ;
}

DWORD * DibMaskPtr (HDIB hdib)
{
    PDIBSTRUCT pdib = hdib ;

    if (!DibIsValid (hdib))
        return 0 ;

    return pdib -> ds.dsBitfields ;
}

void * DibBitsPtr (HDIB hdib)
{
    if (!DibIsValid (hdib))
        return NULL ;

    return ((PDIBSTRUCT) hdib) -> pBits ;
}

/* ---------------------------
    DibSetColor: Obtains entry from the DIB color table
   ---------------------------*/

BOOL DibGetColor (HDIB hdib, int index, RGBQUAD * prgb)
{
    PDIBSTRUCT  pdib = hdib ;
    HDC         hdcMem ;
    int         iReturn ;

    if (!DibIsValid (hdib))
        return 0 ;

    hdcMem = CreateCompatibleDC (NULL) ;
    SelectObject (hdcMem, pdib -> hBitmap) ;
    iReturn = GetDIBColorTable (hdcMem, index, 1, prgb) ;
    DeleteDC (hdcMem) ;
    return iReturn ? TRUE : FALSE ;
```

```
}

/*----------------------
    DibGetColor: Sets an entry in the DIB color table
   ----------------------*/

BOOL DibSetColor (HDIB hdib, int index, RGBQUAD * prgb)
{
    PDIBSTRUCT  pdib = hdib ;
    HDC         hdcMem ;
    int         iReturn ;

    if (!DibIsValid (hdib))
        return 0 ;

    hdcMem = CreateCompatibleDC (NULL) ;
    SelectObject (hdcMem, pdib -> hBitmap) ;
    iReturn = SetDIBColorTable (hdcMem, index, 1, prgb) ;
    DeleteDC (hdcMem) ;

    return iReturn ? TRUE : FALSE ;
}
```

DIBHELP.C 中的大部分函数是自解释的。DibIsValid 函数能有助于保护整个系统。在试图引用 DIBSTRUCT 中的信息之前，其他函数都调用 DibIsValid。所有这些函数都有(而且通常是只有)HDIB 的首参数，(我们将立即看到)该参数在 DIBHELP.H 中定义为空指针。这些函数可以将此参数保存到 PDIBSTRUCT，然后再访问结构中的字段。

注意返回 BOOL 值的 DibIsAddressable 函数。DibIsNotCompressed 函数也可以调用此函数。返回值表示一个个的 DIB 像素能否被编址。

以 DibInfoHeaderSize 开始的函数集将获得 DIB 块中不同组件出现在紧缩 DIB 中的大小。与我们所看到的一样，这些函数有助于将 DIB 块转换成紧缩 DIB，并保存 DIB 文件。这些函数的后面是获得指向不同 DIB 组件的指针的函数集。

尽管 DIBHELP.C 包括名称为 DibInfoHeaderPtr 的函数，而且该函数将获得指向 BITMAPINFOHEADER 结构的指针，但还是没有获得 BITMAPINFO 结构指针的函数——即 DIB 颜色表跟随的信息结构。这是因为在处理 DIB 块时，应用程序并不直接访问这种类型的结构。BITMAPINFOHEADER 结构和颜色掩码都在 DIBSECTION 结构中有效，而且从 CreateDIBSection 函数返回指向像素位的指针，这时通过调用 GetDIBColorTable 和 SetDIBColorTable，就只能间接访问 DIB 颜色表。这些函数都压缩到 DIBHELP 的 DibGetColor 和 DibSetColor 函数。

在 DIBHELP.C 的后面，文件 DibCopyToInfo 分配一个指向 BITMAPINFO 结构的指针，并填充信息，但是那与获得指向内存中现存结构的指针不完全相同。

16.4.3 读、写像素

应用程序维护紧缩 DIB 或 DIB 块的一个引人注目的优点是能够直接操作 DIB 像素位。程序 16-20 所示的 DIBHELP.C 第二部分显示了提供此功能的函数。

程序 16-20 DIBHELP.C 文件的第二部分

DIBHELP.C (第二部分)

```
/*-----------------------------
    DibPixelPtr: Returns a pointer to the pixel at position (x, y)
   -----------------------------*/

BYTE * DibPixelPtr (HDIB hdib, int x, int y)
{
    if (!DibIsAddressable (hdib))
        return NULL ;

    if (x < 0 || x >= DibWidth (hdib) || y < 0 || y >= DibHeight (hdib))
        return NULL ;
    return (((PDIBSTRUCT) hdib) -> ppRow)[y] + (x * DibBitCount (hdib) >> 3) ;
}


/*-----------------------------
    DibGetPixel: Obtains a pixel value at (x, y)
   -----------------------------*/

DWORD DibGetPixel (HDIB hdib, int x, int y)
{
    PBYTE pPixel ;

    if (!(pPixel = DibPixelPtr (hdib, x, y)))
        return 0 ;
    switch (DibBitCount (hdib))
    {
    case  1: return 0x01 & (* pPixel >> (7 - (x & 7))) ;
    case  4: return 0x0F & (* pPixel >> (x & 1 ? 0 : 4)) ;
    case  8: return * pPixel ;
    case 16: return * (WORD *) pPixel ;
    case 24: return 0x00FFFFFF & * (DWORD *) pPixel ;
    case 32: return * (DWORD *) pPixel ;
    }
    return 0 ;
}
```

```
/*-----------------
    DibSetPixel: Sets a pixel value at (x, y)
  -----------------*/

BOOL DibSetPixel (HDIB hdib, int x, int y, DWORD dwPixel)
{
    PBYTE pPixel ;

    if (!(pPixel = DibPixelPtr (hdib, x, y)))
        return FALSE ;

    switch (DibBitCount (hdib))
    {
    case  1: * pPixel &= ~(1      << (7 - (x & 7))) ;
             * pPixel |= dwPixel  << (7 - (x & 7)) ;
             break ;

    case  4: * pPixel &= 0x0F     << (x & 1 ? 4 : 0) ;
             * pPixel |= dwPixel  << (x & 1 ? 0 : 4) ;
             break ;

    case  8: * pPixel = (BYTE) dwPixel ;
             break ;

    case 16: * (WORD *) pPixel = (WORD) dwPixel ;
             break ;
    case 24: * (RGBTRIPLE *) pPixel = * (RGBTRIPLE *) &dwPixel ;
             break ;

    case 32: * (DWORD *) pPixel = dwPixel ;
             break ;
    default:
        return FALSE ;
    }
    return TRUE ;
}

/*-----------------------
    DibGetPixelColor: Obtains the pixel color at (x, y)
  -----------------------*/

BOOL DibGetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb)
{
    DWORD       dwPixel ;
    int         iBitCount ;
    PDIBSTRUCT  pdib = hdib ;

        // Get bit count; also use this as a validity check
```

```
    if (0 == (iBitCount = DibBitCount (hdib)))
        return FALSE ;

        // Get the pixel value

    dwPixel = DibGetPixel (hdib, x, y) ;

        // If the bit-count is 8 or less, index the color table

    if (iBitCount <= 8)
       return DibGetColor (hdib, (int) dwPixel, prgb) ;

       // If the bit-count is 24, just use the pixel

    else if (iBitCount == 24)
    {
         * (RGBTRIPLE *) prgb = * (RGBTRIPLE *) & dwPixel ;
         prgb -> rgbReserved = 0 ;
    }

        // If the bit-count is 32 and the biCompression field is BI_RGB,
        //    just use the pixel

    else if (iBitCount == 32 &&
           pdib -> ds.dsBmih.biCompression == BI_RGB)
    {
         * prgb = * (RGBQUAD *) & dwPixel ;
    }

        // Otherwise, use the mask and shift values
        //   (for best performance, don't use DibMask and DibShift functions)
    else
    {
         prgb -> rgbRed = (BYTE) (((pdib -> ds.dsBitfields[0] & dwPixel)
                                   >> pdib -> iRShift[0]) << pdib -> iLShift[0]) ;

         prgb -> rgbGreen = (BYTE) (((pdib -> ds.dsBitfields[1] & dwPixel)
                                     >> pdib -> iRShift[1]) << pdib -> iLShift[1]) ;

         prgb -> rgbBlue = (BYTE) (((pdib -> ds.dsBitfields[2] & dwPixel)
                                   >> pdib -> iRShift[2]) << pdib -> iLShift[2]) ;
    }
    return TRUE ;
}

/* - - - - - - - - - - - - - - - - - - -
    DibSetPixelColor: Sets the pixel color at (x, y)
   - - - - - - - - - - - - - - - - - - - - */
```

```
BOOL DibSetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb)
{
     DWORD        dwPixel ;
     int          iBitCount ;
     PDIBSTRUCT   pdib = hdib ;

          // Don't do this function for DIBs with color tables

     iBitCount = DibBitCount (hdib) ;

     if (iBitCount <= 8)
          return FALSE ;

          // The rest is just the opposite of DibGetPixelColor

     else if (iBitCount == 24)
     {
          * (RGBTRIPLE *) & dwPixel = * (RGBTRIPLE *) prgb ;
          dwPixel &= 0x00FFFFFF ;
     }
     else if (iBitCount == 32 &&
              pdib -> ds.dsBmih.biCompression == BI_RGB)
     {
          * (RGBQUAD *) & dwPixel = * prgb ;
     }

     else
     {
          dwPixel = (((DWORD) prgb -> rgbRed >> pdib -> iLShift[0])
                         << pdib -> iRShift[0]) ;

          dwPixel |= (((DWORD) prgb -> rgbGreen >> pdib -> iLShift[1])
                         << pdib -> iRShift[1]) ;

          dwPixel |= (((DWORD) prgb -> rgbBlue >> pdib -> iLShift[2])
                         << pdib -> iRShift[2]) ;
     }

     DibSetPixel (hdib, x, y, dwPixel) ;
     return TRUE ;
}
```

这部分 DIBHELP.C 从 DibPixelPtr 函数开始，该函数获得指向保存(或部分保存)有特殊像素的字节的指针。重新调用 DIBSTRUCT 结构的 ppRow 字段，是指向 DIB 中像素行地址的指针，从顶行开始。这样，

```
((PDIBSTRUCT) hdib) -> pprow)[0]
```

就是指向 DIB 顶行最左端像素的指针,而

```
(((PDIBSTRUCT) hdib) -> ppRow)[y] + (x * DibBitCount (hdib) >> 3)
```

是指向位于(x,y)的像素的指针。注意:如果 DIB 不可编址(即已被压缩),或者如果函数的 x 和 y 参数是负数或指向 DIB 外面的区域,则函数将返回 NULL。此检查减慢了函数(和所有依赖于 DibPixelPtr 的函数)的运行速度,下面我将讲述一些更快的例程。

文件后面的 DibGetPixel 和 DibSetPixel 函数利用了 DibPixelPtr。对于 8 位、16 位和 32 位 DIB,这些函数只记录指向合适数据尺寸的指针,并访问像素值。对于 1 位和 4 位的 DIB,则需要掩码和移位。

DibGetColor 函数按 RGBQUAD 结构获得像素颜色。对于 1 位、4 位和 8 位 DIB,这包括使用像素值来从 DIB 颜色表获得颜色。对于 16 位、24 位和 32 位 DIB,通常必须将像素值掩蔽和移位以得到 RGB 颜色。DibSetPixel 函数则相反,它允许从 RGBQUAD 结构设置像素值。该函数只为 16 位、24 位和 32 位 DIB 定义。

16.4.4 创建和转换

程序 16-21 所示的 DIBHELP 第三部分和最后部分演示了如何创建 DIB 块,以及如何将 DIB 块与紧缩 DIB 相互转换。

程序 16-21 DIBHELP.C 文件的第三部分

DIBHELP.C (第三部分)

```
/*-------------------------
     Calculating shift values from color masks is required by the
       DibCreateFromInfo function.
   -------------------------*/

static int MaskToRShift (DWORD dwMask)
{
     int iShift ;

     if (dwMask == 0)
          return 0 ;

     for (iShift = 0 ; !(dwMask & 1) ; iShift ++)
          dwMask >>= 1 ;

     return iShift ;
}

static int MaskToLShift (DWORD dwMask)
{
```

```
    int iShift ;

    if (dwMask == 0)
        return 0 ;

    while (!(dwMask & 1))
        dwMask >> = 1 ;

    for (iShift = 0 ; dwMask & 1 ; iShift++ )
        dwMask >> = 1 ;

    return 8 - iShift ;
}
/*----------------------------------------
    DibCreateFromInfo: All DIB creation functions ultimately call this one.
      This function is responsible for calling CreateDIBSection, allocating
      memory for DIBSTRUCT, and setting up the row pointer.
    ----------------------------------------*/

HDIB DibCreateFromInfo (BITMAPINFO * pbmi)
{
    BYTE        * pBits ;
    DIBSTRUCT   * pdib ;
    HBITMAP       hBitmap ;
    int           i, iRowLength, cy, y ;

    hBitmap = CreateDIBSection (NULL, pbmi, DIB_RGB_COLORS, &pBits, NULL, 0) ;

    if (hBitmap == NULL)
        return NULL ;

    if (NULL == (pdib = malloc (sizeof (DIBSTRUCT))))
    {
        DeleteObject (hBitmap) ;
        return NULL ;
    }

    pdib -> iSignature = HDIB_SIGNATURE ;
    pdib -> hBitmap    = hBitmap ;
    pdib -> pBits      = pBits ;

    GetObject (hBitmap, sizeof (DIBSECTION), &pdib -> ds) ;
        // Notice that we can now use the DIB information functions
        //   defined above.

        // If the compression is BI_BITFIELDS, calculate shifts from masks
    if (DibCompression (pdib) == BI_BITFIELDS)
    {
```

```
        for (i = 0 ; i < 3 ; i++)
        {
            pdib -> iLShift[i] = MaskToLShift (pdib -> ds.dsBitfields[i]) ;
            pdib -> iRShift[i] = MaskToRShift (pdib -> ds.dsBitfields[i]) ;
        }
    }
    // If the compression is BI_RGB, but bit-count is 16 or 32,
    // set the bitfields and the masks
    else if (DibCompression (pdib) == BI_RGB)
    {
        if (DibBitCount (pdib) == 16)
        {
            pdib -> ds.dsBitfields[0] = 0x00007C00 ;
            pdib -> ds.dsBitfields[1] = 0x000003E0 ;
            pdib -> ds.dsBitfields[2] = 0x0000001F ;

            pdib -> iRShift[0] = 10 ;
            pdib -> iRShift[1] =  5 ;
            pdib -> iRShift[2] =  0 ;

            pdib -> iLShift[0] =  3 ;
            pdib -> iLShift[1] =  3 ;
            pdib -> iLShift[2] =  3 ;
        }
        else if (DibBitCount (pdib) == 24 || DibBitCount (pdib) == 32)
        {
            pdib -> ds.dsBitfields[0] = 0x00FF0000 ;
            pdib -> ds.dsBitfields[1] = 0x0000FF00 ;
            pdib -> ds.dsBitfields[2] = 0x000000FF ;

            pdib -> iRShift[0] = 16 ;
            pdib -> iRShift[1] =  8 ;
            pdib -> iRShift[2] =  0 ;

            pdib -> iLShift[0] =  0 ;
            pdib -> iLShift[1] =  0 ;
            pdib -> iLShift[2] =  0 ;
        }
    }
    // Allocate an array of pointers to each row in the DIB

    cy = DibHeight (pdib) ;

    if (NULL == (pdib -> ppRow = malloc (cy * sizeof (BYTE *))))
    {
        free (pdib) ;
        DeleteObject (hBitmap) ;
        return NULL ;
```

```
    }

        // Initialize them.

    iRowLength = DibRowLength (pdib) ;
    if (pbmi -> bmiHeader.biHeight > 0)          // ie, bottom up
    {
        for (y = 0 ; y < cy ; y++)
            pdib -> ppRow[y] = pBits + (cy - y - 1) * iRowLength ;
    }
    else                                        // top down
    {
        for (y = 0 ; y < cy ; y++)
            pdib -> ppRow[y] = pBits + y * iRowLength ;
    }
    return pdib ;
}

/* - - - - - - - - - - - - - - - - - - - - -
    DibDelete: Frees all memory for the DIB section
   - - - - - - - - - - - - - - - - - - - - - */

BOOL DibDelete (HDIB hdib)
{
    DIBSTRUCT * pdib = hdib ;

    if (!DibIsValid (hdib))
        return FALSE ;

    free (pdib -> ppRow) ;
    DeleteObject (pdib -> hBitmap) ;
    free (pdib) ;
    return TRUE ;
}

/* - - - - - - - - - - - - - - - - - - - - -
    DibCreate: Creates an HDIB from explicit arguments
   - - - - - - - - - - - - - - - - - - - - - */

HDIB DibCreate (int cx, int cy, int cBits, int cColors)
{
    BITMAPINFO * pbmi ;
    DWORD        dwInfoSize ;
    HDIB         hDib ;
    int          cEntries ;

    if (cx <= 0 || cy <= 0 ||
       ((cBits != 1) && (cBits != 4) && (cBits != 8) &&
        (cBits != 16) && (cBits != 24) && (cBits != 32)))
```

```
    {
        return NULL ;
    }

    if (cColors != 0)
        cEntries = cColors ;
    else if (cBits <= 8)
        cEntries = 1 << cBits ;

    dwInfoSize = sizeof (BITMAPINFOHEADER) + (cEntries - 1) * sizeof (RGBQUAD);

    if (NULL == (pbmi = malloc (dwInfoSize)))
    {
        return NULL ;
    }

    ZeroMemory (pbmi, dwInfoSize) ;

    pbmi -> bmiHeader.biSize          = sizeof (BITMAPINFOHEADER) ;
    pbmi -> bmiHeader.biWidth         = cx ;
    pbmi -> bmiHeader.biHeight        = cy ;
    pbmi -> bmiHeader.biPlanes        = 1 ;
    pbmi -> bmiHeader.biBitCount      = cBits ;
    pbmi -> bmiHeader.biCompression   = BI_RGB ;
    pbmi -> bmiHeader.biSizeImage     = 0 ;
    pbmi -> bmiHeader.biXPelsPerMeter = 0 ;
    pbmi -> bmiHeader.biYPelsPerMeter = 0 ;
    pbmi -> bmiHeader.biClrUsed       = cColors ;
    pbmi -> bmiHeader.biClrImportant  = 0 ;

    hDib = DibCreateFromInfo (pbmi) ;
    free (pbmi) ;

    return hDib ;
}

/* - - - - - - - - - - - - - - - - - - - - - - -
    DibCopyToInfo: Builds BITMAPINFO structure.
                   Used by DibCopy and DibCopyToDdb
   - - - - - - - - - - - - - - - - - - - - - - - */

static BITMAPINFO * DibCopyToInfo (HDIB hdib)
{
    BITMAPINFO * pbmi ;
    int          i, iNumColors ;
    RGBQUAD    * prgb ;
    if (!DibIsValid (hdib))
        return NULL ;
```

```
        // Allocate the memory

    if (NULL == (pbmi = malloc (DibInfoSize (hdib))))
        return NULL ;

        // Copy the information header

    CopyMemory (pbmi, DibInfoHeaderPtr (hdib),
                                    sizeof (BITMAPINFOHEADER));

        // Copy the possible color masks

    prgb = (RGBQUAD *) ((BYTE *) pbmi + sizeof (BITMAPINFOHEADER)) ;

    if (DibMaskSize (hdib))
    {
         CopyMemory (prgb, DibMaskPtr (hdib), 3 * sizeof (DWORD)) ;

         prgb = (RGBQUAD *) ((BYTE *) prgb + 3 * sizeof (DWORD)) ;
    }
        // Copy the color table

    iNumColors = DibNumColors (hdib) ;

    for (i = 0 ; i < iNumColors ; i++)
        DibGetColor (hdib, i, prgb + i) ;

    return pbmi ;
}

/*-------------------------------------
    DibCopy: Creates a new DIB section from an existing DIB section,
      possibly swapping the DIB width and height.
   -------------------------------------*/

HDIB DibCopy (HDIB hdibSrc, BOOL fRotate)
{
    BITMAPINFO * pbmi ;
    BYTE       * pBitsSrc, * pBitsDst ;
    HDIB         hdibDst ;

    if (!DibIsValid (hdibSrc))
        return NULL ;
    if (NULL == (pbmi = DibCopyToInfo (hdibSrc)))
        return NULL ;

    if (fRotate)
    {
        pbmi->bmiHeader.biWidth = DibHeight (hdibSrc) ;
```

```
        pbmi -> bmiHeader.biHeight = DibWidth (hdibSrc) ;
    }

    hdibDst = DibCreateFromInfo (pbmi) ;
    free (pbmi) ;

    if (hdibDst == NULL)
        return NULL ;

        // Copy the bits

    if (!fRotate)
    {
        pBitsSrc = DibBitsPtr (hdibSrc) ;
        pBitsDst = DibBitsPtr (hdibDst) ;

        CopyMemory (pBitsDst, pBitsSrc, DibBitsSize (hdibSrc)) ;
    }
    return hdibDst ;
}

/*-----------------------------------
    DibCopyToPackedDib is generally used for saving DIBs and for
      transferring DIBs to the clipboard. In the second case, the second
      argument should be set to TRUE so that the memory is allocated
      with the GMEM_SHARE flag.
    -----------------------------------*/

BITMAPINFO * DibCopyToPackedDib (HDIB hdib, BOOL fUseGlobal)
{
    BITMAPINFO * pPackedDib ;
    BYTE       * pBits ;
    DWORD        dwDibSize ;
    HDC          hdcMem ;
    HGLOBAL      hGlobal ;
    int          iNumColors ;
    PDIBSTRUCT   pdib = hdib ;
    RGBQUAD    * prgb ;

    if (!DibIsValid (hdib))
        return NULL ;
        // Allocate memory for packed DIB

    dwDibSize = DibTotalSize (hdib) ;

    if (fUseGlobal)
    {
        hGlobal = GlobalAlloc (GHND | GMEM_SHARE, dwDibSize) ;
```

```
        pPackedDib = GlobalLock (hGlobal) ;
    }
    else
    {
        pPackedDib = malloc (dwDibSize) ;
    }

    if (pPackedDib == NULL)
        return NULL ;

        // Copy the information header

    CopyMemory (pPackedDib, &pdib -> ds.dsBmih, sizeof (BITMAPINFOHEADER)) ;

    prgb = (RGBQUAD *) ((BYTE *) pPackedDib + sizeof (BITMAPINFOHEADER)) ;

        // Copy the possible color masks

    if (pdib -> ds.dsBmih.biCompression == BI_BITFIELDS)
    {
        CopyMemory (prgb, pdib -> ds.dsBitfields, 3 * sizeof (DWORD)) ;

        prgb = (RGBQUAD *) ((BYTE *) prgb + 3 * sizeof (DWORD)) ;
    }
        // Copy the color table

    if (iNumColors = DibNumColors (hdib))
    {
        hdcMem = CreateCompatibleDC (NULL) ;
        SelectObject (hdcMem, pdib -> hBitmap) ;
        GetDIBColorTable (hdcMem, 0, iNumColors, prgb) ;
        DeleteDC (hdcMem) ;
    }

    pBits = (BYTE *) (prgb + iNumColors) ;

        // Copy the bits

    CopyMemory (pBits, pdib -> pBits, DibBitsSize (pdib)) ;
        // If last argument is TRUE, unlock global memory block and
        //     cast it to pointer in preparation for return

    if (fUseGlobal)
    {
        GlobalUnlock (hGlobal) ;
        pPackedDib = (BITMAPINFO *) hGlobal ;
    }
    return pPackedDib ;
```

```
/* ---------------------------------
    DibCopyFromPackedDib is generally used for pasting DIBs from the
      clipboard.
   ---------------------------------*/

HDIB DibCopyFromPackedDib (BITMAPINFO * pPackedDib)
{
    BYTE        * pBits ;
    DWORD         dwInfoSize, dwMaskSize, dwColorSize ;
    int           iBitCount ;
    PDIBSTRUCT    pdib ;

        // Get the size of the information header and do validity check

    dwInfoSize = pPackedDib -> bmiHeader.biSize ;

    if (dwInfoSize != sizeof (BITMAPCOREHEADER) &&
       dwInfoSize != sizeof (BITMAPINFOHEADER) &&
       dwInfoSize != sizeof (BITMAPV4HEADER) &&
       dwInfoSize != sizeof (BITMAPV5HEADER))
    {
        return NULL ;
    }
        // Get the possible size of the color masks

    if (dwInfoSize == sizeof (BITMAPINFOHEADER) &&
        pPackedDib -> bmiHeader.biCompression == BI_BITFIELDS)
    {
        dwMaskSize = 3 * sizeof (DWORD) ;
    }
    else
    {
        dwMaskSize = 0 ;
    }
        // Get the size of the color table
    if (dwInfoSize == sizeof (BITMAPCOREHEADER))
    {
        iBitCount = ((BITMAPCOREHEADER * ) pPackedDib) -> bcBitCount ;

        if (iBitCount <= 8)
        {
            dwColorSize = (1 << iBitCount) * sizeof (RGBTRIPLE) ;
        }
        else
            dwColorSize = 0 ;
    }
```

```
    else                    // all non-OS/2 compatible DIBs
    {
        if (pPackedDib -> bmiHeader.biClrUsed > 0)
        {
            dwColorSize = pPackedDib -> bmiHeader.biClrUsed * sizeof (RGBQUAD);
        }
        else if (pPackedDib -> bmiHeader.biBitCount <= 8)
        {
            dwColorSize = (1 << pPackedDib -> bmiHeader.biBitCount) *
                                                    sizeof (RGBQUAD) ;
        }
        else
        {
            dwColorSize = 0 ;
        }
    }
        // Finally, get the pointer to the bits in the packed DIB

    pBits = (BYTE * ) pPackedDib + dwInfoSize + dwMaskSize + dwColorSize ;

        // Create the HDIB from the packed-DIB pointer

    pdib = DibCreateFromInfo (pPackedDib) ;

        // Copy the pixel bits

    CopyMemory (pdib -> pBits, pBits, DibBitsSize (pdib)) ;

    return pdib ;
}

/* - - - - - - - - - - - - - - - - - - - - - -
    DibFileLoad: Creates a DIB section from a DIB file
   - - - - - - - - - - - - - - - - - - - - - - */

HDIB DibFileLoad (const TCHAR * szFileName)
{
    BITMAPFILEHEADER  bmfh ;
    BITMAPINFO      * pbmi ;
    BOOL              bSuccess ;
    DWORD             dwInfoSize, dwBitsSize, dwBytesRead ;
    HANDLE            hFile ;
    HDIB              hDib ;

        // Open the file: read access, prohibit write access

    hFile = CreateFile (szFileName, GENERIC_READ, FILE_SHARE_READ, NULL,
                        OPEN_EXISTING, FILE_FLAG_SEQUENTIAL_SCAN, NULL) ;
```

```
if (hFile == INVALID_HANDLE_VALUE)
    return NULL ;

    // Read in the BITMAPFILEHEADER

bSuccess = ReadFile (hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                     &dwBytesRead, NULL) ;

if (!bSuccess || (dwBytesRead != sizeof (BITMAPFILEHEADER))
            || (bmfh.bfType != * (WORD *) "BM"))
{
     CloseHandle (hFile) ;
     return NULL ;
}
    // Allocate memory for the information structure & read it in

dwInfoSize = bmfh.bfOffBits - sizeof (BITMAPFILEHEADER) ;

if (NULL == (pbmi = malloc (dwInfoSize)))

{
     CloseHandle (hFile) ;
     return NULL ;
}

bSuccess = ReadFile (hFile, pbmi, dwInfoSize, &dwBytesRead, NULL) ;

if (!bSuccess || (dwBytesRead != dwInfoSize))
{
     CloseHandle (hFile) ;
     free (pbmi) ;
     return NULL ;
}
    // Create the DIB

hDib = DibCreateFromInfo (pbmi) ;
free (pbmi) ;

if (hDib == NULL)
{
     CloseHandle (hFile) ;
     return NULL ;
}
    // Read in the bits

dwBitsSize = bmfh.bfSize - bmfh.bfOffBits ;

bSuccess = ReadFile (hFile, ((PDIBSTRUCT) hDib) -> pBits,
```

```
                        dwBitsSize, &dwBytesRead, NULL) ;
    CloseHandle (hFile) ;

    if (!bSuccess || (dwBytesRead != dwBitsSize))
    {
        DibDelete (hDib) ;
        return NULL ;
    }
    return hDib ;
}

/*-----------------
    DibFileSave: Saves a DIB section to a file
   -----------------*/

BOOL DibFileSave (HDIB hdib, const TCHAR * szFileName)
{
    BITMAPFILEHEADER  bmfh ;
    BITMAPINFO      * pbmi ;
    BOOL              bSuccess ;
    DWORD             dwTotalSize, dwBytesWritten ;
    HANDLE            hFile ;

    hFile = CreateFile (szFileName, GENERIC_WRITE, 0, NULL,
                        CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL) ;
    if (hFile == INVALID_HANDLE_VALUE)
        return FALSE ;

    dwTotalSize         = DibTotalSize (hdib) ;
    bmfh.bfType         = * (WORD *) "BM" ;
    bmfh.bfSize         = sizeof (BITMAPFILEHEADER) + dwTotalSize ;
    bmfh.bfReserved1    = 0 ;
    bmfh.bfReserved2    = 0 ;
    bmfh.bfOffBits      = bmfh.bfSize - DibBitsSize (hdib) ;

        // Write the BITMAPFILEHEADER

    bSuccess = WriteFile (hFile, &bmfh, sizeof (BITMAPFILEHEADER),
                          &dwBytesWritten, NULL) ;

    if (!bSuccess || (dwBytesWritten != sizeof (BITMAPFILEHEADER)))
    {
        CloseHandle (hFile) ;
        DeleteFile (szFileName) ;
        return FALSE ;
    }
        // Get entire DIB in packed-DIB format
```

```
    if (NULL == (pbmi = DibCopyToPackedDib (hdib, FALSE)))
    {
         CloseHandle (hFile) ;
         DeleteFile (szFileName) ;
         return FALSE ;
    }
        // Write out the packed DIB

    bSuccess = WriteFile (hFile, pbmi, dwTotalSize, &dwBytesWritten, NULL) ;
    CloseHandle (hFile) ;
    free (pbmi) ;

    if (!bSuccess || (dwBytesWritten != dwTotalSize))
    {
         DeleteFile (szFileName) ;
         return FALSE ;
    }
    return TRUE ;
}

/*-----------------------
    DibCopyToDdb: For more efficient screen displays
   -----------------------*/

HBITMAP DibCopyToDdb (HDIB hdib, HWND hwnd, HPALETTE hPalette)
{
    BITMAPINFO * pbmi ;
    HBITMAP      hBitmap ;
    HDC          hdc ;

    if (!DibIsValid (hdib))
        return NULL ;

    if (NULL == (pbmi = DibCopyToInfo (hdib)))
        return NULL ;

    hdc = GetDC (hwnd) ;

    if (hPalette)
    {
         SelectPalette (hdc, hPalette, FALSE) ;
         RealizePalette (hdc) ;
    }

    hBitmap = CreateDIBitmap (hdc, DibInfoHeaderPtr (hdib), CBM_INIT,
                              DibBitsPtr (hdib), pbmi, DIB_RGB_COLORS) ;

    ReleaseDC (hwnd, hdc) ;
```

```
    free (pbmi) ;

    return hBitmap ;
}
```

DIBHELP.C 文件的这部分从两个小函数开始，这两个函数根据 16 位和 32 位 DIB 的颜色掩码得到左、右移位值。这些函数在 15.1.7“颜色掩蔽”节中已作说明。

DibCreateFromInfo 函数是 DIBHELP 中唯一调用 CreateDIBSection 并为 DIBSTRUCT 结构分配内存的函数。其他所有创建和复制函数都重复此函数。DibCreateFromInfo 唯一的参数是指向 BITMAPINFO 结构的指针。此结构的颜色表必须存在，但是它不必用有效的值填充。调用 CreateDIBSection 之后，该函数将初始化 DIBSTRUCT 结构的所有字段。注意，在设置 DIBSTRUCT 结构的 ppRow 字段的值时（指向 DIB 行地址的指针），DIB 按从下向上和从上向下的逻辑单独存在。ppRow 的第一个参数总是 DIB 的顶行。

DibDelete 删除 DibCreateFromInfo 中创建的位图，同时释放在该函数中分配的内存。

DibCreate 可能比 DibCreateFromInfo 更像一个从应用程序调用的函数。前三个参数提供像素的宽度、高度和每像素的位数。最后一个参数可以设置为 0（用于默认尺寸的颜色表），或者设置为非 0（用于更小的颜色表并暗指其位数）。

DibCopy 函数根据现存的 DIB 块创建新的 DIB 块，并用 DibCreateInfo 函数为 BITMAPINFO 结构分配内存，并填充所有的数据。DibCopy 函数的一个 BOOL 型参数指出是否在创建新的 DIB 时交换了 DIB 的宽度和高度。我们将在后面看到此函数的用法。

DibCopyToPackedDib 和 DibCopyFromPackedDib 函数的使用通常与通过剪贴板传递 DIB 相关。DibFileLoad 函数从 DIB 文件创建 DIB 节；DibFileSave 函数将数据保存到 DIB 文件。

最后，DibCopyToDdb 函数根据 DIB 创建 GDI 位图对象。注意，该函数需要当前调色板的句柄和程序窗口的句柄。程序窗口句柄用于获得选进并实现调色板的设备描述表。只有这样函数才可以调用 CreateDIBitmap。这在本章前面的 SHOWDIB7 中演示。

16.4.5　DIBHELP 头文件和宏

DIBHELP.H 头文件如程序 16-22 所示。

程序 16-22　DIBHELP.H 文件

DIBHELP.H

```
/*------------------
    DIBHELP.H header file for DIBHELP.C
  ------------------*/

typedef void * HDIB ;
```

```
    // Functions in DIBHELP.C

BOOL DibIsValid (HDIB hdib) ;
HBITMAP DibBitmapHandle (HDIB hdib) ;
int DibWidth (HDIB hdib) ;
int DibHeight (HDIB hdib) ;
int DibBitCount (HDIB hdib) ;
int DibRowLength (HDIB hdib) ;
int DibNumColors (HDIB hdib) ;
DWORD DibMask (HDIB hdib, int i) ;
int DibRShift (HDIB hdib, int i) ;
int DibLShift (HDIB hdib, int i) ;
int DibCompression (HDIB hdib) ;
BOOL DibIsAddressable (HDIB hdib) ;
DWORD DibInfoHeaderSize (HDIB hdib) ;
DWORD DibMaskSize (HDIB hdib) ;
DWORD DibColorSize (HDIB hdib) ;
DWORD DibInfoSize (HDIB hdib) ;
DWORD DibBitsSize (HDIB hdib) ;
DWORD DibTotalSize (HDIB hdib) ;
BITMAPINFOHEADER * DibInfoHeaderPtr (HDIB hdib) ;
DWORD * DibMaskPtr (HDIB hdib) ;
void * DibBitsPtr (HDIB hdib) ;
BOOL DibGetColor (HDIB hdib, int index, RGBQUAD * prgb) ;
BOOL DibSetColor (HDIB hdib, int index, RGBQUAD * prgb) ;
BYTE * DibPixelPtr (HDIB hdib, int x, int y) ;
DWORD DibGetPixel (HDIB hdib, int x, int y) ;
BOOL DibSetPixel (HDIB hdib, int x, int y, DWORD dwPixel) ;
BOOL DibGetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb) ;
BOOL DibSetPixelColor (HDIB hdib, int x, int y, RGBQUAD * prgb) ;
HDIB DibCreateFromInfo (BITMAPINFO * pbmi) ;
BOOL DibDelete (HDIB hdib) ;
HDIB DibCreate (int cx, int cy, int cBits, int cColors) ;
HDIB DibCopy (HDIB hdibSrc, BOOL fRotate) ;
BITMAPINFO * DibCopyToPackedDib (HDIB hdib, BOOL fUseGlobal) ;
HDIB DibCopyFromPackedDib (BITMAPINFO * pPackedDib) ;
HDIB DibFileLoad (const TCHAR * szFileName) ;
BOOL DibFileSave (HDIB hdib, const TCHAR * szFileName) ;
HBITMAP DibCopyToDdb (HDIB hdib, HWND hwnd, HPALETTE hPalette) ;
HDIB DibCreateFromDdb (HBITMAP hBitmap) ;

/* - - - - - - - - - - - - - - - - - - -
     Quickie no-bounds-checked pixel gets and sets
   - - - - - - - - - - - - - - - - - - - */

# define DibPixelPtr1(hdib, x, y) (((* (PBYTE * *) hdib) [y]) + ((x) >> 3))
# define DibPixelPtr4(hdib, x, y) (((* (PBYTE * *) hdib) [y]) + ((x) >> 1))
# define DibPixelPtr8(hdib, x, y) (((* (PBYTE * *) hdib) [y]) + (x) )
```

```
#define DibPixelPtr16(hdib, x, y) \
                ((WORD *)(((* (PBYTE **) hdib)[y]) + (x) * 2))
#define DibPixelPtr 24(hdib, x, y) \
              ((RGBTRIPLE *)(((* (PBYTE **) hdib)[y]) + (x) * 3))
#define DibPixelPtr32(hdib, x, y) \
                ((DWORD *)(((* (PBYTE **) hdib)[y]) + (x) * 4))
#define DibGetPixel1(hdib, x, y) \
          (0x01 & (* DibPixelPtr1 (hdib, x, y) >> (7 - ((x) & 7))))

#define DibGetPixel4(hdib, x, y) \
          (0x0F & (* DibPixelPtr4 (hdib, x, y) >> ((x) & 1 ? 0 : 4)))

#define DibGetPixel8(hdib, x, y)     (* DibPixelPtr8 (hdib, x, y))
#define DibGetPixel16(hdib, x, y)    (* DibPixelPtr16 (hdib, x, y))
#define DibGetPixel24(hdib, x, y)    (* DibPixelPtr24 (hdib, x, y))
#define DibGetPixel32(hdib, x, y)    (* DibPixelPtr32(hdib, x, y))

#define DibSetPixel1(hdib, x, y, p)                                    \
        ((* DibPixelPtr1 (hdib, x, y) &= ~(1 << (7 - ((x) & 7)))),     \
         (* DibPixelPtr1 (hdib, x, y) |= ((p) << (7 - ((x) & 7)))))
#define DibSetPixel4(hdib, x, y, p)                                    \
        ((* DibPixelPtr4 (hdib, x, y) &= (0x0F << ((x) & 1 ? 4 : 0))), \
         (* DibPixelPtr4 (hdib, x, y) |= ((p) << ((x) & 1 ? 0 : 4))))
#define DibSetPixel8(hdib, x, y, p) (* DibPixelPtr8 (hdib, x, y) = p)
#define DibSetPixel16(hdib, x, y, p) (* DibPixelPtr16 (hdib, x, y) = p)
#define DibSetPixel24(hdib, x, y, p) (* DibPixelPtr24 (hdib, x, y) = p)
#define DibSetPixel32(hdib, x, y, p) (* DibPixelPtr32 (hdib, x, y) = p)
```

上述头文件将 HDIB 定义为空指针。应用程序的确不需要了解 HDIB 所指结构的内部结构。此头文件还包括 DIBHELP.C 中所有函数的说明，还有一些宏——非常特殊的宏。

如果再看一看 DIBHELP.C 中的 DibPixelPtr、DibGetPixel 和 DibSetPixel 函数，并试图提高它们的性能，那么您将看到两种可能的解决方法。第一种，可以删除所有的保护，并相信应用程序不会用无效参数调用函数。还可以删除一些函数调用，例如 DibBitCount，并使用指向 DIBSTRUCT 结构内部的指针来直接获得信息。

另一个提高一点性能的方法是删除所有包括每像素位数的逻辑，同时分离每种 DIB 函数——例如 DibGetPixel1、DibGetPixel4、DibGetPixel8 等等。下一个优化是整个删除函数调用，并在内联函数或宏中合并逻辑。

DIBHELP.H 采用宏的方法。它基于 DibPixelPtr、DibGetPixel 和 DibSetPixel 函数提出了三套宏。这些宏都明确对应于特殊的位数。

16.4.6 DIBBLE 程序

DIBBLE 程序，如程序 16-23 所示，用 DIBHELP 函数和宏工作。尽管 DIBBLE 是本书中最

长的程序，但它确实只是一些外理工作的粗略示例，这些外理工作可以在简单的数字图像处理程序中找到。对 DIBBLE 的明显改进是转换成了多文档界面（MDI：multiple document interface），我们将在第十九章学习有关多文档界面的知识。

程序 16-23　DIBBLE 程序

DIBBLE.C

```
/*------------------------
   DIBBLE.C——Bitmap and Palette Program
             (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"
#include "dibconv.h"
#include "resource.h"

#define WM_USER_SETSCROLLS     (WM_USER + 1)
#define WM_USER_DELETEDIB      (WM_USER + 2)
#define WM_USER_DELETEPAL      (WM_USER + 3)
#define WM_USER_CREATEPAL      (WM_USER + 4)

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("Dibble") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HACCEL      hAccel ;
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;
```

```
    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, szAppName,
                         WS_OVERLAPPEDWINDOW | WM_VSCROLL | WM_HSCROLL,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    hAccel = LoadAccelerators (hInstance, szAppName) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        if (! TranslateAccelerator (hwnd, hAccel, &msg))
        {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
    }
    return msg.wParam ;
}

/*----------------------------------------
    DisplayDib: Displays or prints DIB actual size or stretched
                depending on menu selection
  ----------------------------------------*/

int DisplayDib (HDC hdc, HBITMAP hBitmap, int x, int y,
                int cxClient, int cyClient,
                WORD wShow, BOOL fHalftonePalette)
{
    BITMAP  bitmap ;
    HDC     hdcMem ;
    int     cxBitmap, cyBitmap, iReturn ;
    GetObject (hBitmap, sizeof (BITMAP), &bitmap) ;
    cxBitmap = bitmap.bmWidth ;
    cyBitmap = bitmap.bmHeight ;

    SaveDC (hdc) ;

    if (fHalftonePalette)
```

```
        SetStretchBltMode (hdc, HALFTONE) ;
    else
        SetStretchBltMode (hdc, COLORONCOLOR) ;

    hdcMem = CreateCompatibleDC (hdc) ;
    SelectObject (hdcMem, hBitmap) ;

    switch (wShow)
    {
    case IDM_SHOW_NORMAL:
        if (fHalftonePalette)
            iReturn = StretchBlt (hdc,      0, 0,
                                            min (cxClient, cxBitmap - x),
                                            min (cyClient, cyBitmap - y),
                                  hdcMem, x, y,
                                            min (cxClient, cxBitmap - x),
                                            min (cyClient, cyBitmap - y),
                                  SRCCOPY);
        else
            iReturn = BitBlt (hdc,      0, 0,
                                        min (cxClient, cxBitmap - x),
                                        min (cyClient, cyBitmap - y),
                              hdcMem, x, y, SRCCOPY) ;
        break ;

    case IDM_SHOW_CENTER:
        if (fHalftonePalette)
            iReturn = StretchBlt (hdc, (cxClient - cxBitmap) / 2,
                                       (cyClient - cyBitmap) / 2,
                                       cxBitmap, cyBitmap,
                                  hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCOPY);
        else
            iReturn = BitBlt (hdc, (cxClient - cxBitmap) / 2,
                                   (cyClient - cyBitmap) / 2,
                                   cxBitmap, cyBitmap,
                              hdcMem, 0, 0, SRCCOPY) ;
        break ;
    case IDM_SHOW_STRETCH:
        iReturn = StretchBlt (hdc,      0, 0, cxClient, cyClient,
                              hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
        break ;

    case IDM_SHOW_ISOSTRETCH:
        SetMapMode (hdc, MM_ISOTROPIC) ;
        SetWindowExtEx (hdc, cxBitmap, cyBitmap, NULL) ;
        SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;
        SetWindowOrgEx (hdc, cxBitmap / 2, cyBitmap / 2, NULL) ;
        SetViewportOrgEx (hdc, cxClient / 2, cyClient / 2, NULL) ;
```

```
            iReturn = StretchBlt (hdc,      0, 0, cxBitmap, cyBitmap,
                                  hdcMem, 0, 0, cxBitmap, cyBitmap, SRCCOPY) ;
            break ;
        }
        DeleteDC (hdcMem) ;
        RestoreDC (hdc, -1) ;
        return iReturn ;
}

/*-------------------------------
    DibFlipHorizontal: Calls non-optimized DibSetPixel and DibGetPixel
   -------------------------------*/

HDIB DibFlipHorizontal (HDIB hdibSrc)
{
    HDIB hdibDst ;
    int  cx, cy, x, y ;

    if (!DibIsAddressable (hdibSrc))
        return NULL ;

    if (NULL == (hdibDst = DibCopy (hdibSrc, FALSE)))
        return NULL ;

    cx = DibWidth (hdibSrc) ;
    cy = DibHeight (hdibSrc) ;

    for (x = 0 ; x < cx ; x++)
    for (y = 0 ; y < cy ; y++)
    {
        DibSetPixel (hdibDst, x, cy - 1 - y, DibGetPixel (hdibSrc, x, y)) ;
    }
    return hdibDst ;
}

/*-------------------------------
    DibRotateRight: Calls optimized DibSetPixelx and DibGetPixelx
   -------------------------------*/

HDIB DibRotateRight (HDIB hdibSrc)
{
    HDIB hdibDst ;
    int  cx, cy, x, y ;

    if (!DibIsAddressable (hdibSrc))
        return NULL ;

    if (NULL == (hdibDst = DibCopy (hdibSrc, TRUE)))
```

```
    return NULL ;

cx = DibWidth (hdibSrc) ;
cy = DibHeight (hdibSrc) ;

switch (DibBitCount (hdibSrc))
{
case 1:
    for (x = 0 ; x < cx ; x++ )
    for (y = 0 ; y < cy ; y++ )
        DibSetPixel1 (hdibDst, cy - y - 1, x,
            DibGetPixel1 (hdibSrc, x, y)) ;
    break ;

case 4:
    for (x = 0 ; x < cx ; x++ )
    for (y = 0 ; y < cy ; y++ )
        DibSetPixel4 (hdibDst, cy - y - 1, x,
            DibGetPixel4 (hdibSrc, x, y)) ;
    break ;

case 8:
    for (x = 0 ; x < cx ; x++ )
    for (y = 0 ; y < cy ; y++ )
        DibSetPixel8 (hdibDst, cy - y - 1, x,
            DibGetPixel8 (hdibSrc, x, y)) ;
    break ;

case 16:
    for (x = 0 ; x < cx ; x++ )
    for (y = 0 ; y < cy ; y++ )
        DibSetPixel16 (hdibDst, cy - y - 1, x,
            DibGetPixel16 (hdibSrc, x, y)) ;
    break ;

case 24:
    for (x = 0 ; x < cx ; x++ )
    for (y = 0 ; y < cy ; y++ )
        DibSetPixel24 (hdibDst, cy - y - 1, x,
            DibGetPixel24 (hdibSrc, x, y)) ;
    break ;

case 32:
    for (x = 0 ; x < cx ; x++ )
    for (y = 0 ; y < cy ; y++ )
        DibSetPixel32 (hdibDst, cy - y - 1, x,
            DibGetPixel32 (hdibSrc, x, y)) ;
    break ;
}
```

```
    return hdibDst ;
}

/*-----------------------------
    PaletteMenu: Uncheck and check menu item on palette menu
   -----------------------------*/

void PaletteMenu (HMENU hMenu, WORD wItemNew)
{
    static WORD wItem = IDM_PAL_NONE ;

    CheckMenuItem (hMenu, wItem, MF_UNCHECKED) ;
    wItem = wItemNew ;
    CheckMenuItem (hMenu, wItem, MF_CHECKED) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BOOL             fHalftonePalette ;
    static DOCINFO          di = { sizeof (DOCINFO), TEXT ("Dibble: Printing") } ;
    static HBITMAP          hBitmap ;
    static HDIB             hdib ;
    static HMENU            hMenu ;
    static HPALETTE         hPalette ;
    static int              cxClient, cyClient, iVscroll, iHscroll ;
    static OPENFILENAME     ofn ;
    static PRINTDLG         printdlg = { sizeof (PRINTDLG) } ;
    static TCHAR            szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
    static TCHAR            szFilter[] = TEXT ("Bitmap Files (*.BMP)\0*.bmp\0")
                                         TEXT ("All Files (*.*)\0*.*\0\0") ;
    static TCHAR      *     szCompression[] = {
                              TEXT ("BI_RGB"), TEXT ("BI_RLE8"), TEXT ("BI_RLE4"),
                              TEXT ("BI_BITFIELDS"), TEXT ("Unknown") } ;
    static WORD             wShow = IDM_SHOW_NORMAL ;
    BOOL                    fSuccess ;
    BYTE                    *pGlobal ;
    HDC                     hdc, hdcPrn ;
    HGLOBAL                 hGlobal ;
    HDIB                    hdibNew ;
    int                     iEnable, cxPage, cyPage, iConvert ;
    PAINTSTRUCT             ps ;
    SCROLLINFO              si ;
    TCHAR                   szBuffer [256] ;

    switch (message)
    {
    case WM_CREATE:
```

```
        // Save the menu handle in a static variable

    hMenu = GetMenu (hwnd) ;

        // Initialize the OPENFILENAME structure for the File Open
        //    and File Save dialog boxes.

    ofn.lStructSize       = sizeof (OPENFILENAME) ;
    ofn.hwndOwner         = hwnd ;
    ofn.hInstance         = NULL ;
    ofn.lpstrFilter       = szFilter ;
    ofn.lpstrCustomFilter = NULL ;
    ofn.nMaxCustFilter    = 0 ;
    ofn.nFilterIndex      = 0 ;
    ofn.lpstrFile         = szFileName ;
    ofn.nMaxFile          = MAX_PATH ;
    ofn.lpstrFileTitle    = szTitleName ;
    ofn.nMaxFileTitle     = MAX_PATH ;
    ofn.lpstrInitialDir   = NULL ;
    ofn.lpstrTitle        = NULL ;
    ofn.Flags             = OFN_OVERWRITEPROMPT ;
    ofn.nFileOffset       = 0 ;
    ofn.nFileExtension    = 0 ;
    ofn.lpstrDefExt       = TEXT ("bmp") ;
    ofn.lCustData         = 0 ;
    ofn.lpfnHook          = NULL ;
    ofn.lpTemplateName    = NULL ;
    return 0 ;

case WM_DISPLAYCHANGE:
    SendMessage (hwnd, WM_USER_DELETEPAL, 0, 0) ;
    SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;
    return 0 ;

case WM_SIZE:
        // Save the client area width and height in static variables.

    cxClient = LOWORD (lParam) ;
    cyClient = HIWORD (lParam) ;

    wParam = FALSE ;
                                    // fall through

        // WM_USER_SETSCROLLS: Programmer-defined Message!
        // Set the scroll bars. If the display mode is not normal,
        // make them invisible. If wParam is TRUE, reset the
        // scroll bar position.
```

```
case WM_USER_SETSCROLLS:
    if (hdib == NULL || wShow != IDM_SHOW_NORMAL)
    {
        si.cbSize = sizeof (SCROLLINFO) ;
        si.fMask = SIF_RANGE ;
        si.nMin  = 0 ;
        si.nMax  = 0 ;
        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
        SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
    }
    else
    {
            // First the vertical scroll

        si.cbSize = sizeof (SCROLLINFO) ;
        si.fMask  = SIF_ALL ;

        GetScrollInfo (hwnd, SB_VERT, &si) ;
        si.nMin  = 0 ;
        si.nMax  = DibHeight (hdib) ;
        si.nPage = cyClient ;
        if ((BOOL) wParam)
            si.nPos = 0 ;

        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_VERT, &si) ;

        iVscroll = si.nPos ;

            // Then the horizontal scroll

        GetScrollInfo (hwnd, SB_HORZ, &si) ;
        si.nMin  = 0 ;
        si.nMax  = DibWidth (hdib) ;
        si.nPage = cxClient ;

        if ((BOOL) wParam)
            si.nPos = 0 ;

        SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
        GetScrollInfo (hwnd, SB_HORZ, &si) ;

        iHscroll = si.nPos ;
    }
    return 0 ;
    // WM_VSCROLL: Vertically scroll the DIB
```

```
case WM_VSCROLL:
     si.cbSize = sizeof (SCROLLINFO) ;
     si.fMask = SIF_ALL ;
     GetScrollInfo (hwnd, SB_VERT, &si) ;

     iVscroll = si.nPos ;

     switch (LOWORD (wParam))
     {
     case SB_LINEUP:          si.nPos -= 1 ;              break ;
     case SB_LINEDOWN:        si.nPos += 1 ;              break ;
     case SB_PAGEUP:          si.nPos -= si.nPage ;       break ;
     case SB_PAGEDOWN:        si.nPos += si.nPage ;       break ;
     case SB_THUMBTRACK:      si.nPos  = si.nTrackPos ;   break ;
     default:                                             break ;
     }
     si.fMask = SIF_POS ;
     SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
     GetScrollInfo (hwnd, SB_VERT, &si) ;
     if (si.nPos != iVscroll)
     {
          ScrollWindow (hwnd, 0, iVscroll - si.nPos, NULL, NULL) ;
          iVscroll = si.nPos ;
          UpdateWindow (hwnd) ;
     }
     return 0 ;

     // WM_HSCROLL: Horizontally scroll the DIB
case WM_HSCROLL:
     si.cbSize = sizeof (SCROLLINFO) ;
     si.fMask = SIF_ALL ;
     GetScrollInfo (hwnd, SB_HORZ, &si) ;

     iHscroll = si.nPos ;

     switch (LOWORD (wParam))
     {
     case SB_LINELEFT:        si.nPos -= 1 ;              break ;
     case SB_LINERIGHT:       si.nPos += 1 ;              break ;
     case SB_PAGELEFT:        si.nPos -= si.nPage ;       break ;
     case SB_PAGERIGHT:       si.nPos += si.nPage ;       break ;
     case SB_THUMBTRACK:      si.nPos = si.nTrackPos ;    break ;
     default:                                             break ;
     }

     si.fMask = SIF_POS ;
```

```
            SetScrollInfo (hwnd, SB_HORZ, &si, TRUE) ;
            GetScrollInfo (hwnd, SB_HORZ, &si) ;

            if (si.nPos != iHscroll)
            {
                 ScrollWindow (hwnd, iHscroll - si.nPos, 0, NULL, NULL) ;
                 iHscroll = si.nPos ;
                 UpdateWindow (hwnd) ;
            }
            return 0 ;

            // WM_INITMENUPOPUP: Enable or Gray menu items

        case WM_INITMENUPOPUP:
            if (hdib)
                iEnable = MF_ENABLED ;
            else
                iEnable = MF_GRAYED ;
            EnableMenuItem (hMenu, IDM_FILE_SAVE,          iEnable) ;
            EnableMenuItem (hMenu, IDM_FILE_PRINT,         iEnable) ;
            EnableMenuItem (hMenu, IDM_FILE_PROPERTIES,    iEnable) ;
            EnableMenuItem (hMenu, IDM_EDIT_CUT,           iEnable) ;
            EnableMenuItem (hMenu, IDM_EDIT_COPY,          iEnable) ;
            EnableMenuItem (hMenu, IDM_EDIT_DELETE,        iEnable) ;

            if (DibIsAddressable (hdib))
                iEnable = MF_ENABLED ;
            else
                iEnable = MF_GRAYED ;

            EnableMenuItem (hMenu, IDM_EDIT_ROTATE,        iEnable) ;
            EnableMenuItem (hMenu, IDM_EDIT_FLIP,          iEnable) ;
            EnableMenuItem (hMenu, IDM_CONVERT_01,         iEnable) ;
            EnableMenuItem (hMenu, IDM_CONVERT_04,         iEnable) ;
            EnableMenuItem (hMenu, IDM_CONVERT_08,         iEnable) ;
            EnableMenuItem (hMenu, IDM_CONVERT_16,         iEnable) ;
            EnableMenuItem (hMenu, IDM_CONVERT_24,         iEnable) ;
            EnableMenuItem (hMenu, IDM_CONVERT_32,         iEnable) ;

            switch (DibBitCount (hdib))
            {
            case  1: EnableMenuItem (hMenu, IDM_CONVERT_01, MF_GRAYED) ; break ;
            case  4: EnableMenuItem (hMenu, IDM_CONVERT_04, MF_GRAYED) ; break ;
            case  8: EnableMenuItem (hMenu, IDM_CONVERT_08, MF_GRAYED) ; break ;
            case 16: EnableMenuItem (hMenu, IDM_CONVERT_16, MF_GRAYED) ; break ;
```

```
        case 24: EnableMenuItem (hMenu, IDM_CONVERT_24, MF_GRAYED) ; break ;
        case 32: EnableMenuItem (hMenu, IDM_CONVERT_32, MF_GRAYED) ; break ;
        }

        if (hdib && DibColorSize (hdib) > 0)
            iEnable = MF_ENABLED ;
        else
            iEnable = MF_GRAYED ;

        EnableMenuItem (hMenu, IDM_PAL_DIBTABLE,     iEnable) ;
        if (DibIsAddressable (hdib) && DibBitCount (hdib) > 8)
            iEnable = MF_ENABLED ;
        else
            iEnable = MF_GRAYED ;

        EnableMenuItem (hMenu, IDM_PAL_OPT_POP4,     iEnable) ;
        EnableMenuItem (hMenu, IDM_PAL_OPT_POP5,     iEnable) ;
        EnableMenuItem (hMenu, IDM_PAL_OPT_POP6,     iEnable) ;
        EnableMenuItem (hMenu, IDM_PAL_OPT_MEDCUT,   iEnable) ;
        EnableMenuItem (hMenu, IDM_EDIT_PASTE,
            IsClipboardFormatAvailable (CF_DIB) ? MF_ENABLED : MF_GRAYED) ;

        return 0 ;
        // WM_COMMAND: Process all menu commands.

    case WM_COMMAND:
        iConvert = 0 ;

        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:

                // Show the File Open dialog box

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing DIB and palette, delete them
            SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;

                // Load the DIB into memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            hdib = DibFileLoad (szFileName) ;
```

```
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                // Reset the scroll bars

            SendMessage (hwnd, WM_USER_SETSCROLLS, TRUE, 0) ;

                // Create the palette and DDB

            SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;

            if (!hdib)
            {
                MessageBox (hwnd, TEXT ("Cannot load DIB file!"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
            }
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case IDM_FILE_SAVE:

                // Show the File Save dialog box

            if (!GetSaveFileName (&ofn))
                return 0 ;

                // Save the DIB to memory

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            fSuccess = DibFileSave (hdib, szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (!fSuccess)
                MessageBox (hwnd, TEXT ("Cannot save DIB file!"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
            return 0 ;

        case IDM_FILE_PRINT:
            if (!hdib)
                return 0 ;

                // Get printer DC
```

```
printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;

if (!PrintDlg (&printdlg))
     return 0 ;

if (NULL == (hdcPrn = printdlg.hDC))
{
     MessageBox (hwnd, TEXT ("Cannot obtain Printer DC"),
                 szAppName, MB_ICONEXCLAMATION | MB_OK) ;
     return 0 ;
}
    // Check if the printer can print bitmaps

if (!(RC_BITBLT & GetDeviceCaps (hdcPrn, RASTERCAPS)))
{
     DeleteDC (hdcPrn) ;
     MessageBox (hwnd, TEXT ("Printer cannot print bitmaps"),
                 szAppName, MB_ICONEXCLAMATION | MB_OK) ;
     return 0 ;
}
    // Get size of printable area of page
cxPage = GetDeviceCaps (hdcPrn, HORZRES) ;
cyPage = GetDeviceCaps (hdcPrn, VERTRES) ;

fSuccess = FALSE ;

    // Send the DIB to the printer

SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
ShowCursor (TRUE) ;

if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
{
     DisplayDib (hdcPrn, DibBitmapHandle (hdib), 0, 0,
                 cxPage, cyPage, wShow, FALSE) ;

     if (EndPage (hdcPrn) > 0)
     {
          fSuccess = TRUE ;
          EndDoc (hdcPrn) ;
     }
}
ShowCursor (FALSE) ;
SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

DeleteDC (hdcPrn) ;

if (!fSuccess)
```

```
            MessageBox (hwnd, TEXT ("Could not print bitmap"),
                        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;

    case IDM_FILE_PROPERTIES:
        if (!hdib)
            return 0 ;

        wsprintf (szBuffer, TEXT ("Pixel width: \t%i\n")
                            TEXT ("Pixel height: \t%i\n")
                            TEXT ("Bits per pixel: \t%i\n")
                            TEXT ("Number of colors: \t%i\n")
                            TEXT ("Compression: \t%s\n"),
                  DibWidth (hdib), DibHeight (hdib),
                  DibBitCount (hdib), DibNumColors (hdib),
                  szCompression [min (3, DibCompression (hdib))]) ;
        MessageBox (hwnd, szBuffer, szAppName,
                    MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;

    case IDM_APP_EXIT:
        SendMessage (hwnd, WM_CLOSE, 0, 0) ;
        return 0 ;

    case IDM_EDIT_COPY:
    case IDM_EDIT_CUT:
        if (!(hGlobal = DibCopyToPackedDib (hdib, TRUE)))
            return 0 ;
        OpenClipboard (hwnd) ;
        EmptyClipboard () ;
        SetClipboardData (CF_DIB, hGlobal) ;
        CloseClipboard () ;

        if (LOWORD (wParam) == IDM_EDIT_COPY)
            return 0 ;
                                        // fall through for IDM_EDIT_CUT
    case IDM_EDIT_DELETE:
        SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;
        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

    case IDM_EDIT_PASTE:
        OpenClipboard (hwnd) ;

        hGlobal = GetClipboardData (CF_DIB) ;
        pGlobal = GlobalLock (hGlobal) ;
```

```
            // If there's an existing DIB and palette, delete them.
            // Then convert the packed DIB to an HDIB.

        if (pGlobal)
        {
            SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;
            hdib = DibCopyFromPackedDib ((BITMAPINFO * ) pGlobal) ;
            SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;
        }

        GlobalUnlock (hGlobal) ;
        CloseClipboard () ;

            // Reset the scroll bars

        SendMessage (hwnd, WM_USER_SETSCROLLS, TRUE, 0) ;
        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

    case IDM_EDIT_ROTATE:
        if (hdibNew = DibRotateRight (hdib))
        {
            DibDelete (hdib) ;
            DeleteObject (hBitmap) ;
            hdib = hdibNew ;
            hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
            SendMessage (hwnd, WM_USER_SETSCROLLS, TRUE, 0) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
        }
        else
        {
            MessageBox (hwnd, TEXT ("Not enough memory"),
                        szAppName, MB_OK | MB_ICONEXCLAMATION) ;
        }
        return 0 ;

    case IDM_EDIT_FLIP:
        if (hdibNew = DibFlipHorizontal (hdib))
        {
            DibDelete (hdib) ;
            DeleteObject (hBitmap) ;
            hdib = hdibNew ;
            hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
        }
        else
        {
            MessageBox (hwnd, TEXT ("Not enough memory"),
```

```
                    szAppName, MB_OK | MB_ICONEXCLAMATION) ;
        }
        return 0 ;

    case IDM_SHOW_NORMAL:
    case IDM_SHOW_CENTER:
    case IDM_SHOW_STRETCH:
    case IDM_SHOW_ISOSTRETCH:
        CheckMenuItem (hMenu, wShow, MF_UNCHECKED) ;
        wShow = LOWORD (wParam) ;
        CheckMenuItem (hMenu, wShow, MF_CHECKED) ;
        SendMessage (hwnd, WM_USER_SETSCROLLS, FALSE, 0) ;

        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

    case IDM_CONVERT_32: iConvert += 8 ;
    case IDM_CONVERT_24: iConvert += 8 ;
    case IDM_CONVERT_16: iConvert += 8 ;
    case IDM_CONVERT_08: iConvert += 4 ;
    case IDM_CONVERT_04: iConvert += 3 ;
    case IDM_CONVERT_01: iConvert += 1 ;
        SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
        ShowCursor (TRUE) ;

        hdibNew = DibConvert (hdib, iConvert) ;

        ShowCursor (FALSE) ;
        SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

        if (hdibNew)
        {
            SendMessage (hwnd, WM_USER_DELETEDIB, 0, 0) ;
            hdib = hdibNew ;
            SendMessage (hwnd, WM_USER_CREATEPAL, TRUE, 0) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
        }
        else
        {
            MessageBox (hwnd, TEXT ("Not enough memory"),
                        szAppName, MB_OK | MB_ICONEXCLAMATION) ;
        }
        return 0 ;

    case IDM_APP_ABOUT:
        MessageBox (hwnd, TEXT ("Dibble (c) Charles Petzold, 1998"),
```

```
                szAppName, MB_OK | MB_ICONEXCLAMATION) ;
    return 0 ;
}

    // All the other WM_COMMAND messages are from the palette
    //   items. Any existing palette is deleted, and the cursor
    //   is set to the hourglass.

SendMessage (hwnd, WM_USER_DELETEPAL, 0, 0) ;
SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
ShowCursor (TRUE) ;
    // Notice that all messages for palette items are ended
    //   with break rather than return. This is to allow
    //   additional processing later on.

switch (LOWORD (wParam))
{
case IDM_PAL_DIBTABLE:
    hPalette = DibPalDibTable (hdib) ;
    break ;

case IDM_PAL_HALFTONE:
    hdc = GetDC (hwnd) ;

    if (hPalette = CreateHalftonePalette (hdc))
        fHalftonePalette = TRUE ;

    ReleaseDC (hwnd, hdc) ;
    break ;

case IDM_PAL_ALLPURPOSE:
    hPalette = DibPalAllPurpose () ;
    break ;

case IDM_PAL_GRAY2:   hPalette = DibPalUniformGrays (   2) ; break ;
case IDM_PAL_GRAY3:   hPalette = DibPalUniformGrays (   3) ; break ;
case IDM_PAL_GRAY4:   hPalette = DibPalUniformGrays (   4) ; break ;
case IDM_PAL_GRAY8:   hPalette = DibPalUniformGrays (   8) ; break ;
case IDM_PAL_GRAY16:  hPalette = DibPalUniformGrays ( 16) ; break ;
case IDM_PAL_GRAY32:  hPalette = DibPalUniformGrays ( 32) ; break ;
case IDM_PAL_GRAY64:  hPalette = DibPalUniformGrays ( 64) ; break ;
case IDM_PAL_GRAY128: hPalette = DibPalUniformGrays (128) ; break ;
case IDM_PAL_GRAY256: hPalette = DibPalUniformGrays (256) ; break ;

case IDM_PAL_RGB222: hPalette = DibPalUniformColors (2,2,2); break;
case IDM_PAL_RGB333: hPalette = DibPalUniformColors (3,3,3); break;
```

```
            case IDM_PAL_RGB444: hPalette = DibPalUniformColors (4,4,4); break;
            case IDM_PAL_RGB555: hPalette = DibPalUniformColors (5,5,5); break;
            case IDM_PAL_RGB666: hPalette = DibPalUniformColors (6,6,6); break;
            case IDM_PAL_RGB775: hPalette = DibPalUniformColors (7,7,5); break;
            case IDM_PAL_RGB757: hPalette = DibPalUniformColors (7,5,7); break;
            case IDM_PAL_RGB577: hPalette = DibPalUniformColors (5,7,7); break;
            case IDM_PAL_RGB884: hPalette = DibPalUniformColors (8,8,4); break;
            case IDM_PAL_RGB848: hPalette = DibPalUniformColors (8,4,8); break;
            case IDM_PAL_RGB488: hPalette = DibPalUniformColors (4,8,8); break;

            case IDM_PAL_OPT_POP4:
                hPalette = DibPalPopularity (hdib, 4) ;
                break ;
            case IDM_PAL_OPT_POP5:
                hPalette = DibPalPopularity (hdib, 5) ;
                break ;

            case IDM_PAL_OPT_POP6:
                hPalette = DibPalPopularity (hdib, 6) ;
                break ;

            case IDM_PAL_OPT_MEDCUT:
                hPalette = DibPalMedianCut (hdib, 6) ;
                break ;
            }

                // After processing Palette items from the menu, the cursor
                //    is restored to an arrow, the menu item is checked, and
                //    the window is invalidated.

            hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (hPalette)
                PaletteMenu (hMenu, (LOWORD (wParam))) ;

            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

            // This programmer-defined message deletes an existing DIB
            //    in preparation for getting a new one. Invoked during
            //    File Open command, Edit Paste command, and others.

        case WM_USER_DELETEDIB:
            if (hdib)
```

```
        {
            DibDelete (hdib) ;
            hdib = NULL ;
        }
        SendMessage (hwnd, WM_USER_DELETEPAL, 0, 0) ;
        return 0 ;

        // This programmer-defined message deletes an existing palette
        //    in preparation for defining a new one.

    case WM_USER_DELETEPAL:
        if (hPalette)
        {
            DeleteObject (hPalette) ;
            hPalette = NULL ;
            fHalftonePalette = FALSE ;
            PaletteMenu (hMenu, IDM_PAL_NONE) ;
        }
        if (hBitmap)
            DeleteObject (hBitmap) ;

        return 0 ;

        // Programmer-defined message to create a new palette based on
        //    a new DIB. If wParam == TRUE, create a DDB as well.

    case WM_USER_CREATEPAL:
        if (hdib)
        {
            hdc = GetDC (hwnd) ;

            if (!(RC_PALETTE & GetDeviceCaps (hdc, RASTERCAPS)))
            {
                PaletteMenu (hMenu, IDM_PAL_NONE) ;
            }
            else if (hPalette = DibPalDibTable (hdib))
            {
                PaletteMenu (hMenu, IDM_PAL_DIBTABLE) ;
            }
            else if (hPalette = CreateHalftonePalette (hdc))
            {
                fHalftonePalette = TRUE ;
                PaletteMenu (hMenu, IDM_PAL_HALFTONE) ;
            }
            ReleaseDC (hwnd, hdc) ;

            if ((BOOL) wParam)
```

```
                hBitmap = DibCopyToDdb (hdib, hwnd, hPalette) ;
            }
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            if (hPalette)
            {
                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;
            }
            if (hBitmap)
            {
                DisplayDib (hdc,
                            fHalftonePalette ? DibBitmapHandle (hdib) : hBitmap,
                            iHscroll, iVscroll,
                            cxClient, cyClient,
                            wShow, fHalftonePalette) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_QUERYNEWPALETTE:
            if (!hPalette)
                return FALSE ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            InvalidateRect (hwnd, NULL, TRUE) ;

            ReleaseDC (hwnd, hdc) ;
            return TRUE ;

        case WM_PALETTECHANGED:
            if (!hPalette || (HWND) wParam == hwnd)
                break ;

            hdc = GetDC (hwnd) ;
            SelectPalette (hdc, hPalette, FALSE) ;
            RealizePalette (hdc) ;
            UpdateColors (hdc) ;

            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
```

```
            if (hdib)
                DibDelete (hdib) ;

            if (hBitmap)
                DeleteObject (hBitmap) ;

            if (hPalette)
                DeleteObject (hPalette) ;

            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

DIBBLE.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

DIBBLE MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open...\tCtrl+O",          IDM_FILE_OPEN
        MENUITEM "&Save...\tCtrl+S",          IDM_FILE_SAVE
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",         IDM_FILE_PRINT
        MENUITEM SEPARATOR
        MENUITEM "Propert&ies...",            IDM_FILE_PROPERTIES
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                     IDM_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "Cu&t\tCtrl+X",              IDM_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",             IDM_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",            IDM_EDIT_PASTE
        MENUITEM "&Delete\tDelete",           IDM_EDIT_DELETE
        MENUITEM SEPARATOR
        MENUITEM "&Flip",                     IDM_EDIT_FLIP
        MENUITEM "&Rotate",                   IDM_EDIT_ROTATE
    END
```

```
    POPUP "&Show"
    BEGIN
        MENUITEM "&Actual Size",                IDM_SHOW_NORMAL, CHECKED
        MENUITEM "&Center",                     IDM_SHOW_CENTER
        MENUITEM "&Stretch to Window",          IDM_SHOW_STRETCH
        MENUITEM "Stretch &Isotropically",      IDM_SHOW_ISOSTRETCH
    END
    POPUP "&Palette"
    BEGIN
        MENUITEM "&None",                       IDM_PAL_NONE, CHECKED
        MENUITEM "&Dib ColorTable",             IDM_PAL_DIBTABLE
        MENUITEM "&Halftone",                   IDM_PAL_HALFTONE
        MENUITEM "&All-Purpose",                IDM_PAL_ALLPURPOSE
        POPUP "&Gray Shades"
        BEGIN
            MENUITEM "&1. 2 Grays",             IDM_PAL_GRAY2
            MENUITEM "&2. 3 Grays",             IDM_PAL_GRAY3
            MENUITEM "&3. 4 Grays",             IDM_PAL_GRAY4
            MENUITEM "&4. 8 Grays",             IDM_PAL_GRAY8
            MENUITEM "&5. 16 Grays",            IDM_PAL_GRAY16
            MENUITEM "&6. 32 Grays",            IDM_PAL_GRAY32
            MENUITEM "&7. 64 Grays",            IDM_PAL_GRAY64
            MENUITEM "&8. 128 Grays",           IDM_PAL_GRAY128
            MENUITEM "&9. 256 Grays",           IDM_PAL_GRAY256
        END
    POPUP "&Uniform Colors"
    BEGIN
        MENUITEM "&1. 2R x 2G x 2B (8)",        IDM_PAL_RGB222
        MENUITEM "&2. 3R x 3G x 3B (27)",       IDM_PAL_RGB333
        MENUITEM "&3. 4R x 4G x 4B (64)",       IDM_PAL_RGB444
        MENUITEM "&4. 5R x 5G x 5B (125)",      IDM_PAL_RGB555
        MENUITEM "&5. 6R x 6G x 6B (216)",      IDM_PAL_RGB666
        MENUITEM "&6. 7R x 7G x 5B (245)",      IDM_PAL_RGB775
        MENUITEM "&7. 7R x 5B x 7B (245)",      IDM_PAL_RGB757
        MENUITEM "&8. 5R x 7G x 7B (245)",      IDM_PAL_RGB577
        MENUITEM "&9. 8R x 8G x 4B (256)",      IDM_PAL_RGB884
        MENUITEM "&A. 8R x 4G x 8B (256)",      IDM_PAL_RGB848
        MENUITEM "&B. 4R x 8G x 8B (256)",      IDM_PAL_RGB488
    END
    POPUP "&Optimized"
    BEGIN
        MENUITEM "&1. Popularity Algorithm (4 bits)", IDM_PAL_OPT_POP4
        MENUITEM "&2. Popularity Algorithm (5 bits)", IDM_PAL_OPT_POP5
        MENUITEM "&3. Popularity Algorithm (6 bits)", IDM_PAL_OPT_POP6
```

```
        MENUITEM "&4. Median Cut Algorithm",        IDM_PAL_OPT_MEDCUT
    END
END
POPUP "Con&vert"
BEGIN
    MENUITEM "&1. to 1 bit per pixel",          IDM_CONVERT_01
    MENUITEM "&2. to 4 bits per pixel",         IDM_CONVERT_04
    MENUITEM "&3. to 8 bits per pixel",         IDM_CONVERT_08
    MENUITEM "&4. to 16 bits per pixel",        IDM_CONVERT_16
    MENUITEM "&5. to 24 bits per pixel",        IDM_CONVERT_24
    MENUITEM "&6. to 32 bits per pixel",        IDM_CONVERT_32
END
POPUP "&Help"
BEGIN
    MENUITEM "&About",                          IDM_APP_ABOUT
END
END

//////////////////////////////////////////////////////////////////////////////
// Accelerator

DIBBLE ACCELERATORS DISCARDABLE
BEGIN
    "C",            IDM_EDIT_COPY,          VIRTKEY, CONTROL, NOINVERT
    "O",            IDM_FILE_OPEN,          VIRTKEY, CONTROL, NOINVERT
    "P",            IDM_FILE_PRINT,         VIRTKEY, CONTROL, NOINVERT
    "S",            IDM_FILE_SAVE,          VIRTKEY, CONTROL, NOINVERT
    "V",            IDM_EDIT_PASTE,         VIRTKEY, CONTROL, NOINVERT
    VK_DELETE,      IDM_EDIT_DELETE,        VIRTKEY, NOINVERT
    "X",            IDM_EDIT_CUT,           VIRTKEY, CONTROL, NOINVERT
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by Dibble.rc

#define IDM_FILE_OPEN                   40001
#define IDM_FILE_SAVE                   40002
#define IDM_FILE_PRINT                  40003
#define IDM_FILE_PROPERTIES             40004
#define IDM_APP_EXIT                    40005
#define IDM_EDIT_CUT                    40006
#define IDM_EDIT_COPY                   40007
#define IDM_EDIT_PASTE                  40008
#define IDM_EDIT_DELETE                 40009
```

```
# define IDM_EDIT_FLIP              40010
# define IDM_EDIT_ROTATE            40011
# define IDM_SHOW_NORMAL            40012
# define IDM_SHOW_CENTER            40013
# define IDM_SHOW_STRETCH           40014
# define IDM_SHOW_ISOSTRETCH        40015
# define IDM_PAL_NONE               40016
# define IDM_PAL_DIBTABLE           40017
# define IDM_PAL_HALFTONE           40018
# define IDM_PAL_ALLPURPOSE         40019
# define IDM_PAL_GRAY2              40020
# define IDM_PAL_GRAY3              40021
# define IDM_PAL_GRAY4              40022
# define IDM_PAL_GRAY8              40023
# define IDM_PAL_GRAY16             40024
# define IDM_PAL_GRAY32             40025
# define IDM_PAL_GRAY64             40026
# define IDM_PAL_GRAY128            40027
# define IDM_PAL_GRAY256            40028
# define IDM_PAL_RGB222             40029
# define IDM_PAL_RGB333             40030
# define IDM_PAL_RGB444             40031
# define IDM_PAL_RGB555             40032
# define IDM_PAL_RGB666             40033
# define IDM_PAL_RGB775             40034
# define IDM_PAL_RGB757             40035
# define IDM_PAL_RGB577             40036
# define IDM_PAL_RGB884             40037
# define IDM_PAL_RGB848             40038
# define IDM_PAL_RGB488             40039
# define IDM_PAL_OPT_POP4           40040
# define IDM_PAL_OPT_POP5           40041
# define IDM_PAL_OPT_POP6           40042
# define IDM_PAL_OPT_MEDCUT         40043
# define IDM_CONVERT_01             40044
# define IDM_CONVERT_04             40045
# define IDM_CONVERT_08             40046
# define IDM_CONVERT_16             40047
# define IDM_CONVERT_24             40048
# define IDM_CONVERT_32             40049
# define IDM_APP_ABOUT              40050
```

DIBBLE 使用了两个其他文件，我将简要地说明它们。DIBCONV 文件(DIBCONV.C

和 DIBCONV.H)在两种不同格式之间转换——例如,从每像素 24 位转换成每像素 8 位。DIBPAL 文件(DIBPAL.C 和 DIBPAL.H)创建调色板。

DIBBLE 维护 WndProc 中的三个重要的静态变量。这些是调用 hdib 的 HDIB 句柄、调用 hPalette 的 HPALETTE 句柄和调用 hBitmap 的 HBITMAP 句柄。HDIB 来自 DIBHELP 中的不同函数;HPALETTE 来自 DIBPAL 中的不同函数或 CreateHalftonePalette 函数;而 HBITMAP 句柄来自 DIBHELP.C 中的 DibCopyToDdb 函数并帮助加速屏幕显示,特别是在 256 色视频模式下。不过,无论在程序创建新的 DIB 块(明显地)时,还是程序创建不同的调色板(不很明显)时,这个句柄都必须重新创建。

让我们从功能而非次序的角度来考察 DIBBLE。

文件加载和保存

DIBBLE 可以在响应 IDM_FILE_LOAD 和 IDM_FILE_SAVE 的 WM_COMMAND 消息过程中加载 DIB 文件并保存这些文件。在处理这些消息期间,DIBBLE 通过分别调用 GetOpenFileName 和 GetSaveFileName 来激活公用文件对话框。

对于 File、Save 菜单命令,DIBBLE 只需调用 DibFileSave。对于 File、Open 菜单命令,DIBBLE 必须首先删除前面的 HDIB、调色板和位图对象。它通过发送一个 WM_USER_DELETEDIB 消息来完成这件事,此消息通过调用 DibDelete 和 DeleteObject 来处理,然后 DIBBLE 调用 DIBHELP 中的 DibFileLoad 函数,发送 WM_USER_SETSCROLLS 和 WM_USER_CREATEPAL 消息来重新设置滚动条,并创建调色板。WM_USER_CREATEPAL 消息也位于程序从 DIB 节创建的新的.DDB 位置。

显示、滚动和打印

DIBBLE 的菜单允许它以实际尺寸在客户区左上角显示 DIB,或者在客户区中间显示 DIB,或者伸展到填充客户区,或者在保持纵横比的情况下尽可能大地填充客户区。您可以通过 DIBBLE 的 Show 菜单来选择需要的选项。注意,这些是最后一章的 SHOWDIB2 程序中 4 个相同的有效选项。

在 WM_PAINT 消息期间——也是处理 File、Print 命令的过程中——DIBBLE 调用 DisplayDib 函数。注意,DisplayDib 使用 BitBlt 和 StretchBlt,而不是使用 SetDIBitsToDevice 和 StretchDIBits。在 WM_PAINT 消息期间,传递给函数的位图句柄由 DibCopyToDdb 函数创建,并在 WM_USER_CREATEPAL 消息期间调用,其中 DDB 与视频设备描述表兼容。当处理 File、Print 命令时,DIBBLE 调用 DisplayDib,其中可用的 DIB 节句柄来自 DIBHELP.C 中的 DibBitmapHandle 函数。

另外要注意,DIBBLE 保留一个被称作 fHalftonePalette 的静态 BOOL 变量,如果从 CreateHalftonePalette 函数中获得 hPalette,则此变量设置为 TRUE。这将迫使 DisplayDib

函数调用 StretchBlt 而不是调用 BitBlt，即使按实际尺寸显示 DIB。fHalftonePalette 变量也导致 WM_PAINT 过程将 DIB 节句柄传递给 DisplayDib 函数，而不是由 DibCopyToDdb 函数创建的位图句柄。本章前面讨论过半色调调色板的使用，并在 SHOWDIB5 程序中进行了演示。

第一次使用示例程序时，DIBBLE 允许在客户区中滚动 DIB。只有按实际尺寸显示 DIB 时，才显示滚动条。在处理 WM_PAINT 时，WndProc 简单地将滚动条的当前位置传递给 DisplayDib 函数。

剪贴板

对于 Cut 和 Copy 菜单项，DIBBLE 调用 DIBHELP 中的 DibCopyToPackedDib 函数。该函数将获得所有的 DIB 组件，并将它们放入大的内存块中。

对于第一次使用本书中的某些示例程序来说，DIBBLE 从剪贴板中粘贴 DIB。这包括调用 DibCopyFromPackedDib 函数，并替换窗口过程前面保存的 HDIB、调色板和位图。

翻转和旋转

DIBBLE 中的 Edit 菜单中除了常见的 Cut、Copy、Paste 和 Delete 选项之外，还包括两个附加项——Flip 和 Rotate。Flip 选项使位图绕水平轴翻转——即上下颠倒翻转。Rotate 选项使位图顺时针旋转 90°。这两个函数都需要通过将它们从一个 DIB 复制到另一个来访问所有的 DIB 像素。（因为这两个函数不需要创建新的调色板，所以不删除和重新创建调色板。）

Flip 菜单选项使用 DibFlipHorizontal 函数，此函数也位于 DIBBLE.C 文件。此函数调用 DibCopy 来获得 DIB 精确的副本，然后进入将原 DIB 中的像素复制到新 DIB 的循环，但是复制这些像素是为了上下翻转图像。注意，此函数调用 DibGetPixel 和 DibSetPixel。这些是 DIBHELP.C 中的通用（但不像我们所希望的那么快）函数。

为了说明 DibGetPixel 和 DibSetPixel 函数与 DIBHELP.H 中运行更快的 DibGetPixel 和 DibSetPixel 宏之间的区别，DibRotateRight 函数使用了宏。然而首先要注意的是，该函数调用 DibCopy 时，第二个参数设置为 TRUE。这导致 DibCopy 翻转原 DIB 的宽度和高度来创建新的 DIB。另外，像素位不能由 DibCopy 函数复制。但是，DibRotateRight 函数有 6 个不同的循环将像素位从原 DIB 复制到新的 DIB——每一个都对应不同的 DIB 像素宽度（1 位、4 位、8 位、16 位、24 位和 32 位）。虽然包括了更多的代码，但是函数更快了。

尽管可以使用 Flip Horizontal 和 Rotate Right 选项来模拟 Flip Vertical、Rotate Left 和 Rotate 180°函数，但通常程序将直接执行所有这些选项。当然，DIBBLE 只是个演示程序而已。

16.4.7　简单调色板和优化调色板

在 DIBBLE 中，您可以在 256 色视频显示器上选择的不同的调色板来显示 DIB。这些都在 DIBBLE 的 Palette 菜单中列出。除了半色调调色板以外，其余的都直接由 Windows 函数调用创建，创建不同调色板的所有函数都由程序 16-24 所示的 DIBPAL 文件提供。

程序 16-24　DIBPAL 文件

DIBPAL.H

```
/*----------------
   DIBPAL.H header file for DIBPAL.C
  ----------------*/

HPALETTE DibPalDibTable (HDIB hdib) ;
HPALETTE DibPalAllPurpose (void) ;
HPALETTE DibPalUniformGrays (int iNum) ;
HPALETTE DibPalUniformColors (int iNumR, int iNumG, int iNumB) ;
HPALETTE DibPalVga (void) ;
HPALETTE DibPalPopularity (HDIB hdib, int iRes) ;
HPALETTE DibPalMedianCut (HDIB hdib, int iRes) ;
```

DIBPAL.C

```
/*----------------
   DIBPAL.C——Palette-Creation Functions
             (c) Charles Petzold, 1998
  ----------------*/

#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"

/*----------------------
   DibPalDibTable: Creates a palette from the DIB color table
  ----------------------*/

HPALETTE DibPalDibTable (HDIB hdib)
{
     HPALETTE        hPalette ;
     int             i, iNum ;
     LOGPALETTE *    plp ;
     RGBQUAD         rgb ;

     if (0 == (iNum = DibNumColors (hdib)))
          return NULL ;
```

```
    plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;

    plp -> palVersion    = 0x0300 ;
    plp -> palNumEntries = iNum ;

    for (i = 0 ; i < iNum ; i++ )
    {
        DibGetColor (hdib, i, &rgb) ;

        plp -> palPalEntry[i].peRed   = rgb.rgbRed ;
        plp -> palPalEntry[i].peGreen = rgb.rgbGreen ;
        plp -> palPalEntry[i].peBlue  = rgb.rgbBlue ;
        plp -> palPalEntry[i].peFlags = 0 ;
    }
    hPalette = CreatePalette (plp) ;
    free (plp) ;
    return hPalette ;
}

/*-------------------------------------
    DibPalAllPurpose: Creates a palette suitable for a wide variety
        of images; the palette has 247 entries, but 15 of them are
        duplicates or match the standard 20 colors.
   -------------------------------------*/

HPALETTE DibPalAllPurpose (void)
{
    HPALETTE        hPalette ;
    int             i, incr, R, G, B ;
    LOGPALETTE *    plp ;

    plp = malloc (sizeof (LOGPALETTE) + 246 * sizeof (PALETTEENTRY)) ;

    plp -> palVersion    = 0x0300 ;
    plp -> palNumEntries = 247 ;

        // The following loop calculates 31 gray shades, but 3 of them
        //     will match the standard 20 colors

    for (i = 0, G = 0, incr = 8 ; G <= 0xFF ; i++, G += incr)
    {
        plp -> palPalEntry[i].peRed   = (BYTE) G ;
        plp -> palPalEntry[i].peGreen = (BYTE) G ;
        plp -> palPalEntry[i].peBlue  = (BYTE) G ;
        plp -> palPalEntry[i].peFlags = 0 ;

        incr = (incr == 9 ? 8 : 9) ;
    }
```

```
        // The following loop is responsible for 216 entries, but 8 of
        //    them will match the standard 20 colors, and another
        //    4 of them will match the gray shades above.

    for (R = 0 ; R <= 0xFF ; R += 0x33)
    for (G = 0 ; G <= 0xFF ; G += 0x33)
    for (B = 0 ; B <= 0xFF ; B += 0x33)
    {
        plp -> palPalEntry[i].peRed   = (BYTE) R ;
        plp -> palPalEntry[i].peGreen = (BYTE) G ;
        plp -> palPalEntry[i].peBlue  = (BYTE) B ;
        plp -> palPalEntry[i].peFlags = 0 ;

        i++ ;
    }
    hPalette = CreatePalette (plp) ;

    free (plp) ;
    return hPalette ;
}

/*-------------------------------------
    DibPalUniformGrays:  Creates a palette of iNum grays, uniformly spaced
   -------------------------------------*/

HPALETTE DibPalUniformGrays (int iNum)
{
    HPALETTE        hPalette ;
    int             i ;
    LOGPALETTE *    plp ;

    plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;

    plp -> palVersion    = 0x0300 ;
    plp -> palNumEntries = iNum ;

    for (i = 0 ; i < iNum ; i++)
    {
        plp -> palPalEntry[i].peRed   =
        plp -> palPalEntry[i].peGreen =
        plp -> palPalEntry[i].peBlue  = (BYTE) (i * 255 / (iNum - 1)) ;
        plp -> palPalEntry[i].peFlags = 0 ;
    }
    hPalette = CreatePalette (plp) ;
    free (plp) ;
    return hPalette ;
}
```

```
/*-------------------------------------
    DibPalUniformColors: Creates a palette of iNumR x iNumG x iNumB colors
   -------------------------------------*/

HPALETTE DibPalUniformColors (int iNumR, int iNumG, int iNumB)
{
    HPALETTE        hPalette ;
    int             i, iNum, R, G, B ;
    LOGPALETTE *    plp ;

    iNum = iNumR * iNumG * iNumB ;

    plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;

    plp -> palVersion    = 0x0300 ;
    plp -> palNumEntries = iNumR * iNumG * iNumB ;

    i = 0 ;
    for (R = 0 ; R < iNumR ; R++)
    for (G = 0 ; G < iNumG ; G++)
    for (B = 0 ; B < iNumB ; B++)
    {
        plp -> palPalEntry[i].peRed   = (BYTE) (R * 255 / (iNumR - 1)) ;
        plp -> palPalEntry[i].peGreen = (BYTE) (G * 255 / (iNumG - 1)) ;
        plp -> palPalEntry[i].peBlue  = (BYTE) (B * 255 / (iNumB - 1)) ;
        plp -> palPalEntry[i].peFlags = 0 ;

        i++ ;
    }
    hPalette = CreatePalette (plp) ;
    free (plp) ;
    return hPalette ;
}

/*-------------------------------------
    DibPalVga: Creates a palette based on standard 16 VGA colors
   -------------------------------------*/

HPALETTE DibPalVga (void)
{
    static RGBQUAD rgb [16] = { 0x00, 0x00, 0x00, 0x00,
                                0x00, 0x00, 0x80, 0x00,
                                0x00, 0x80, 0x00, 0x00,
                                0x00, 0x80, 0x80, 0x00,
                                0x80, 0x00, 0x00, 0x00,
                                0x80, 0x00, 0x80, 0x00,
                                0x80, 0x80, 0x00, 0x00,
                                0x80, 0x80, 0x80, 0x00,
                                0xC0, 0xC0, 0xC0, 0x00,
```

```
                                0x00, 0x00, 0xFF, 0x00,
                                0x00, 0xFF, 0x00, 0x00,
                                0x00, 0xFF, 0xFF, 0x00,
                                0xFF, 0x00, 0x00, 0x00,
                                0xFF, 0x00, 0xFF, 0x00,
                                0xFF, 0xFF, 0x00, 0x00,
                                0xFF, 0xFF, 0xFF, 0x00 } ;
     HPALETTE        hPalette ;
     int             i ;
     LOGPALETTE *    plp ;

     plp = malloc (sizeof (LOGPALETTE) + 15 * sizeof (PALETTEENTRY)) ;
     plp -> palVersion    = 0x0300 ;
     plp -> palNumEntries = 16 ;

     for (i = 0 ; i < 16 ; i++)
     {
          plp -> palPalEntry[i].peRed   = rgb[i].rgbRed ;
          plp -> palPalEntry[i].peGreen = rgb[i].rgbGreen ;
          plp -> palPalEntry[i].peBlue  = rgb[i].rgbBlue ;
          plp -> palPalEntry[i].peFlags = 0 ;
     }
     hPalette = CreatePalette (plp) ;
     free (plp) ;
     return hPalette ;
}

/*-------------------
     Macro used in palette optimization routines
   -------------------*/


#define PACK_RGB(R,G,B,iRes) ((int) (R) | ((int) (G) << (iRes)) |          \
                                          ((int) (B) << ((iRes) + (iRes))))

/*-----------------------------
     AccumColorCounts: Fills up piCount (indexed by a packed RGB color)
      with counts of pixels of that color.
   -----------------------------*/

static void AccumColorCounts (HDIB hdib, int * piCount, int iRes)
{
     int         x, y, cx, cy ;
     RGBQUAD     rgb ;

     cx = DibWidth (hdib) ;
     cy = DibHeight (hdib) ;
```

```
    for (y = 0 ; y < cy ; y++)
    for (x = 0 ; x < cx ; x++)
    {
        DibGetPixelColor (hdib, x, y, &rgb) ;

        rgb.rgbRed   >>= (8 - iRes) ;
        rgb.rgbGreen >>= (8 - iRes) ;
        rgb.rgbBlue  >>= (8 - iRes) ;

        ++piCount [PACK_RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue, iRes)] ;
    }
}

/*-------------------------
    DibPalPopularity: Popularity algorithm for optimized colors
  -------------------------*/

HPALETTE DibPalPopularity (HDIB hdib, int iRes)
{
    HPALETTE        hPalette ;
    int             i, iArraySize, iEntry, iCount, iIndex, iMask, R, G, B ;
    int           * piCount ;
    LOGPALETTE    * plp ;

        // Validity checks

    if (DibBitCount (hdib) < 16)
        return NULL ;

    if (iRes < 3 || iRes > 8)
       return NULL ;

        // Allocate array for counting pixel colors

    iArraySize = 1 << (3 * iRes) ;
    iMask = (1 << iRes) - 1 ;

    if (NULL == (piCount = calloc (iArraySize, sizeof (int))))
        return NULL ;

        // Get the color counts

    AccumColorCounts (hdib, piCount, iRes) ;

        // Set up a palette

    plp = malloc (sizeof (LOGPALETTE) + 235 * sizeof (PALETTEENTRY)) ;
```

```
    plp -> palVersion = 0x0300 ;

    for (iEntry = 0 ; iEntry < 236 ; iEntry++)
    {
        for (i = 0, iCount = 0 ; i < iArraySize ; i++)
            if (piCount[i] > iCount

            {
                iCount = piCount[i] ;
                iIndex = i ;
            }
        if (iCount == 0)
            break ;

        R = (iMask & iIndex                    ) << (8 - iRes) ;
        G = (iMask & (iIndex >>          iRes )) << (8 - iRes) ;
        B = (iMask & (iIndex >> (iRes + iRes))) << (8 - iRes) ;

        plp -> palPalEntry[iEntry].peRed   = (BYTE) R ;
        plp -> palPalEntry[iEntry].peGreen = (BYTE) G ;
        plp -> palPalEntry[iEntry].peBlue  = (BYTE) B ;
        plp -> palPalEntry[iEntry].peFlags = 0 ;

        piCount [iIndex] = 0 ;
    }
        // On exit from the loop iEntry will be the number of stored entries

    plp -> palNumEntries = iEntry ;

        // Create the palette, clean up, and return the palette handle

    hPalette = CreatePalette (plp) ;

    free (piCount) ;
    free (plp) ;

    return hPalette ;
}

/* - - - - - - - - - - - - - - - - - - - - - - -
    Structures used for implementing median cut algorithm
   - - - - - - - - - - - - - - - - - - - - - - - */

typedef struct                  // defines dimension of a box
{
    int Rmin, Rmax, Gmin, Gmax, Bmin, Bmax ;
}
MINMAX ;
```

```
typedef struct                  // for Compare routine for qsort
{
     int        iBoxCount ;
     RGBQUAD rgbBoxAv ;
}

BOXES ;

/* - - - - - - - - - - - - -
     FindAverageColor: In a box
   - - - - - - - - - - - - - */

static int FindAverageColor (int * piCount, MINMAX mm,
                             int iRes, RGBQUAD * prgb)
{
     int R, G, B, iR, iG, iB, iTotal, iCount ;

          // Initialize some variables

     iTotal = iR = iG = iB = 0 ;

          // Loop through all colors in the box

     for (R = mm.Rmin ; R <= mm.Rmax ; R++ )
     for (G = mm.Gmin ; G <= mm.Gmax ; G++ )
     for (B = mm.Bmin ; B <= mm.Bmax ; B++ )
     {
               // Get the number of pixels of that color

          iCount = piCount [PACK_RGB (R, G, B, iRes)] ;

               // Weight the pixel count by the color value

          iR += iCount * R ;
          iG += iCount * G ;
          iB += iCount * B ;

          iTotal += iCount ;
     }
          // Find the average color

     prgb -> rgbRed   = (BYTE) ((iR / iTotal) << (8 - iRes)) ;
     prgb -> rgbGreen = (BYTE) ((iG / iTotal) << (8 - iRes)) ;
     prgb -> rgbBlue  = (BYTE) ((iB / iTotal) << (8 - iRes)) ;

          // Return the total number of pixels in the box

     return iTotal ;
```

```
}

/*------------
     CutBox:  Divide a box in two
   ------------*/

static void CutBox (int * piCount, int iBoxCount, MINMAX mm,
                    int iRes, int iLevel, BOXES * pboxes, int * piEntry)
{
     int       iCount, R, G, B ;
     MINMAX  mmNew ;

          // If the box is empty, return

     if (iBoxCount == 0)
          return ;

          // If the nesting level is 8, or the box is one pixel, we're ready
          // to find the average color in the box and save it along with
          // the number of pixels of that color

     if (iLevel == 8 || (mm.Rmin == mm.Rmax &&
                         mm.Gmin == mm.Gmax &&
                         mm.Bmin == mm.Bmax))
     {
          pboxes[ * piEntry].iBoxCount =
               FindAverageColor (piCount, mm, iRes, &pboxes[ * piEntry].rgbBoxAv) ;

          ( * piEntry) ++ ;
     }
          // Otherwise, if blue is the largest side, split it

     else if ((mm.Bmax - mm.Bmin > mm.Rmax - mm.Rmin) &&
              (mm.Bmax - mm.Bmin > mm.Gmax - mm.Gmin))
     {
               // Initialize a counter and loop through the blue side
          iCount = 0 ;

          for (B = mm.Bmin ; B < mm.Bmax ; B++ )
          {
                    // Accumulate all the pixels for each successive blue value

               for (R = mm.Rmin ; R <= mm.Rmax ; R++ )
               for (G = mm.Gmin ; G <= mm.Gmax ; G++ )
                    iCount += piCount [PACK_RGB (R, G, B, iRes)] ;

                    // If it's more than half the box count, we're there
```

```
                if (iCount >= iBoxCount / 2)
                    break ;

                    // If the next blue value will be the max, we're there

                if (B == mm.Bmax - 1)
                    break ;

            }
                // Cut the two split boxes.
                // The second argument to CutBox is the new box count.
                // The third argument is the new min and max values.

            mmNew = mm ;
            mmNew.Bmin = mm.Bmin ;
            mmNew.Bmax = B ;

            CutBox (piCount, iCount, mmNew, iRes, iLevel + 1,
                    pboxes, piEntry) ;

            mmNew.Bmin = B + 1 ;
            mmNew.Bmax = mm.Bmax ;

            CutBox (piCount, iBoxCount-iCount, mmNew, iRes, iLevel + 1,
                    pboxes, piEntry) ;
        }
        // Otherwise, if red is the largest side, split it (just like blue)

    else if (mm.Rmax - mm.Rmin > mm.Gmax - mm.Gmin)
    {
        iCount = 0 ;

        for (R = mm.Rmin ; R < mm.Rmax ; R++)
        {
            for (B = mm.Bmin ; B <= mm.Bmax ; B++)
            for (G = mm.Gmin ; G <= mm.Gmax ; G++)
                iCount += piCount [PACK_RGB (R, G, B, iRes)] ;

            if (iCount >= iBoxCount / 2)
                break ;

            if (R == mm.Rmax - 1)
                break ;
        }
        mmNew = mm ;
        mmNew.Rmin = mm.Rmin ;
        mmNew.Rmax = R ;
```

```
            CutBox (piCount, iCount, mmNew, iRes, iLevel + 1,
                    pboxes, piEntry) ;

            mmNew.Rmin = R + 1 ;
            mmNew.Rmax = mm.Rmax ;
            CutBox (piCount, iBoxCount - iCount, mmNew, iRes, iLevel + 1,
                    pboxes, piEntry) ;
        }
        // Otherwise, split along the green size
        else
        {
            iCount = 0 ;

            for (G = mm.Gmin ; G < mm.Gmax ; G++)
            {
                for (B = mm.Bmin ; B <= mm.Bmax ; B++)
                for (R = mm.Rmin ; R <= mm.Rmax ; R++)
                    iCount += piCount [PACK_RGB (R, G, B, iRes)] ;

                    if (iCount >= iBoxCount / 2)
                        break ;

                    if (G == mm.Gmax - 1)
                        break ;
                }
                mmNew = mm ;
                mmNew.Gmin = mm.Gmin ;
                mmNew.Gmax = G ;

                CutBox (piCount, iCount, mmNew, iRes, iLevel + 1,
                        pboxes, piEntry) ;

                mmNew.Gmin = G + 1 ;
                mmNew.Gmax = mm.Gmax ;

                CutBox (piCount, iBoxCount - iCount, mmNew, iRes, iLevel + 1,
                        pboxes, piEntry) ;
            }
}

/* - - - - - - - - - - - - -
    Compare routine for qsort
   - - - - - - - - - - - - - */

static int Compare (const BOXES * pbox1, const BOXES * pbox2)
{
    return pbox1 -> iBoxCount - pbox2 -> iBoxCount ;
}
```

```
/*-------------------------------------
   DibPalMedianCut: Creates palette based on median cut algorithm
  -------------------------------------*/

HPALETTE DibPalMedianCut (HDIB hdib, int iRes)
{
     BOXES          boxes [256] ;
     HPALETTE       hPalette ;
     int            i, iArraySize, iCount, R, G, B, iTotCount, iDim, iEntry = 0 ;
     int          * piCount ;
     LOGPALETTE * plp ;
     MINMAX         mm ;

          // Validity checks

     if (DibBitCount (hdib) < 16)
          return NULL ;

     if (iRes < 3 || iRes > 8)
          return NULL ;

          // Accumulate counts of pixel colors

     iArraySize = 1 << (3 * iRes) ;

     if (NULL == (piCount = calloc (iArraySize, sizeof (int))))
          return NULL ;

     AccumColorCounts (hdib, piCount, iRes) ;

          // Find the dimensions of the total box

     iDim = 1 << iRes ;

     mm.Rmin = mm.Gmin = mm.Bmin = iDim - 1 ;
     mm.Rmax = mm.Gmax = mm.Bmax = 0 ;

     iTotCount = 0 ;

     for (R = 0 ; R < iDim ; R++)
     for (G = 0 ; G < iDim ; G++)
     for (B = 0 ; B < iDim ; B++)
          if ((iCount = piCount [PACK_RGB (R, G, B, iRes)]) > 0)
          {
               iTotCount += iCount ;

               if (R < mm.Rmin) mm.Rmin = R ;
               if (G < mm.Gmin) mm.Gmin = G ;
```

```
            if (B < mm.Bmin) mm.Bmin = B ;
            if (R > mm.Rmax) mm.Rmax = R ;
            if (G > mm.Gmax) mm.Gmax = G ;
            if (B > mm.Bmax) mm.Bmax = B ;
        }
        // Cut the first box (iterative function).
        // On return, the boxes structure will have up to 256 RGB values,
        //       one for each of the boxes, and the number of pixels in
        //       each box.
        // The iEntry value will indicate the number of non-empty boxes.
    CutBox (piCount, iTotCount, mm, iRes, 0, boxes, &iEntry) ;
    free (piCount) ;

        // Sort the RGB table by the number of pixels for each color

    qsort (boxes, iEntry, sizeof (BOXES), Compare) ;
    plp = malloc (sizeof (LOGPALETTE) + (iEntry - 1) * sizeof (PALETTEENTRY)) ;

    if (plp == NULL)
        return NULL ;

    plp -> palVersion    = 0x0300 ;
    plp -> palNumEntries = iEntry ;

    for (i = 0 ; i < iEntry ; i++)
    {
        plp -> palPalEntry[i].peRed   = boxes[i].rgbBoxAv.rgbRed ;
        plp -> palPalEntry[i].peGreen = boxes[i].rgbBoxAv.rgbGreen ;
        plp -> palPalEntry[i].peBlue  = boxes[i].rgbBoxAv.rgbBlue ;
        plp -> palPalEntry[i].peFlags = 0 ;
    }

    hPalette = CreatePalette (plp) ;

    free (plp) ;
    return hPalette ;
}
```

第一个函数——DibPalDibTable——看起来应该很熟悉。它根据 DIB 的颜色表创建了调色板。这与本章前面的 SHOWDIB3 中用到的 PACKEDIB.C 里的 PackedDibCreatePalette 函数相似。在 SHOWDIB3 中，只有当 DIB 有颜色表时才运行此函数。在 8 位视频模式下试图显示 16 位、24 位或 32 位 DIB 时，此函数就没用了。

默认情况下，运行在 256 色显示模式下时，DIBBLE 将首先尝试调用 DibPalDibTable 来根据 DIB 颜色表创建调色板。如果 DIB 没有颜色表，则 DIBBLE 将调用 CreateHalftonePalette，并将 fHalftonePalette 变量设置为 TRUE。此逻辑发生在 WM_USER_CREATEPAL 消息期间。

DIBPAL.C 也执行函数 DibPalAllPurpose，因为此函数与 SHOWDIB4 中的 CreateAllPurposePalette 函数非常相似，所以它看起来也很熟悉。您也可以从 DIBBLE 菜单中选择此调色板。

在 256 色模式下显示位图最有意义的是，您可以直接控制 Windows 用于显示图像的颜色。如果您选择并实现调色板，则 Winsows 将使用此调色板中的颜色，而不是其他调色板中的颜色。

例如，您可以用 DibPalUniformGrays 函数来单独创建一种灰色底纹调色板。使用两种灰色底纹的调色板则只含有 00-00-00(黑色)和 FF-FF-FF(白色)。用此调色板来输出图像将提供某些照片中常用的高对比"黑白"效果。使用 3 种灰色底纹将在黑色和白色中间添加中间灰色，使用 4 种灰色底纹将添加 2 种灰色底纹。

用 8 种灰色底纹，您就有可能看到明显的轮廓——相同灰色底纹的无规则斑点，虽然很明显地运行了最接近颜色算法，但是一般仍不带有任何审美判断。通常到 16 种灰色底纹就可以明显改善图像质量。使用 32 种灰色底纹则可以消除某些轮廓线；而目前普遍认为 64 种灰色底纹是现在大多数显示设备的极限。在这点以上，再提高就很勉强了。超过 64 种灰色底纹对于 6 位颜色分辨率的设备没有什么提高。

迄今为止，对于 8 位视频模式下显示 16 位、24 位和 32 位彩色 DIB，我们能做的最好的事情就是设计通用调色板(对于灰色底纹图像微不足道，但是通常不适于彩色图像)或者使用半色调调色板，它用混合色显示来联合通用颜色调色板。

还应注意，当您在 8 位颜色模式下为大的 16 位、24 位或 32 位 DIB 选择通用调色板时，出于显示目的，程序将花费一些时间来根据 DIB 创建 GDI 位图对象。如果没有使用调色板，则程序根据 DIB 来创建 DDB 的时间会更少。(用 8 位彩色模式显示大 24 位 DIB 时，比较 SHOWDIB1 和 SHOWDIB4 的性能，您也能看出这点区别。)这是为什么呢?

它按最接近颜色查找。通常，用 8 位视频模式显示 24 位 DIB(或者将 DIB 转换为 DDB)时，GDI 必须将 DIB 中的每个像素都与 20 种静态颜色中的一种相匹配。完成此操作的唯一方法是决定哪种静态颜色与像素颜色最接近。这包括计算像素与三维 RGB 颜色中每种静态颜色的距离。这将花些时间，特别在 DIB 图像中有上百万个像素时。

在创建 232 色调色板时，例如 DIBBLE 和 SHOWDIB4 中的通用调色板，您会很快将搜索最接近颜色的时间增加到超过 11 倍! GDI 现在必须彻底查找 232 种颜色，而不是 20 种。那就是显示 DIB 的整个工作放慢的原因。

这里的教训是避免在 8 位视频模式下显示 24 位(或 16 位，或 32 位)DIB。您应该找出最接近 DIB 图像颜色范围的 256 色调色板，来将它们转换成 8 位 DIB。这经常被称为"优化调色板"。当我研究这个问题的时候，Paul Heckbert 编写的《Color Image Quantization for Frame Buffer Displays》(刊登在 1982 年 7 月出版的《Computer Graphics》上)对此问题有所帮助。

统一分布

创建 256 色调色板最简单的方法是选择范围统一的 RGB 颜色值，它与 DibPalAllPurpose 中的方法相似。此方法的优点是您不必检查 DIB 中的实际像素。这个函数是 DibPalCreateUniformColors，它创建基于范围统一的 RGB 主索引的调色板。

一个合理的分布包括 8 级的红色和绿色，以及 4 级的蓝色（这是很不容易看出来的）。调色板是 RGB 颜色值的集合，它是红色和绿色值 0x00、0x24、0x49、0x6D、0x92、0xB6、0xDB 和 0xFF，以及蓝色值 0x00、0x55、0xAA 和 0xFF 的所有可能的组合，共有 256 种颜色。另一种可能的统一分布调色板使用 6 级红色、绿色和蓝色。此调色板是红色、绿色和蓝色值为 0x00、0x33、0x66、0x99、0xCC，以及 0xFF 的所有可能的组合，调色板中的颜色数是 6^3，即 216。

这两个选项和其他几个选项都由 DIBBLE 提供。

"Popularity"算法

"Popularity"算法是 256 色调色板问题相当明显的解决方法。您要做的就是遍历位图中的所有像素，并找出 256 种最普通的 RGB 颜色值。这些就是您在调色板中使用的值。它在 DIBPAL 的 DibPalPopularity 函数中实现。

不过，如果每种颜色都使用整个 24 位，而且如果假定需要用整数来计算所有的颜色，那么数组将占据 64 MB 内存。而且，您可以发现位图中实际上没有（或很少）复制 24 位像素值，这样就没有更多的常规颜色。

要解决这个问题，您可以只使用每个红色、绿色和蓝色值中最重要的 n 位——例如，6 位而不是 8 位。因为大多数的彩色扫描仪和视频显示适配器都只有 6 位的分辨率，所以这种规定更有意义。这将数组减少到大小更合理的 256 KB 或 1 MB。只使用 5 位能将可用的颜色总数减少到32 768。通常，使用 5 位要比 6 位的性能更好。对此，您可以用 DIBBLE 和一些图像颜色来自己检验。

"Median Cut"算法

DIBPAL.C 中的 DibPalMedianCut 函数执行 Paul Heckbert 的 Median Cut 算法。此算法在概念上相当简单，但在代码中实现要比 Popularity 算法更困难，它适合递归函数。

画出 RGB 颜色立方体。图像中的每个像素都是此立方体中的一个点。一些点可能代表图像中的多个像素。找出包括图像中所有像素的三维框，找出此框的最大尺寸，并将框分成两个，每个框都包括相同数量的像素。对于这两个框，执行相同的操作。现在就有 4 个框，将这 4 个框分成 8 个，然后再分成 16 个、32 个、64 个、128 个和 256 个。

现在有 256 个框，每个框都包括相同数量的像素。取每个框中像素 RGB 颜色值的平均值，并将结果用于调色板。

实际上，这些框通常包含像素的数量并不相同。例如，通常包括单个点的框会有更多的像素。这发生在黑色和白色上。有时，框根本不用像素结束。如果这样，您就可以分开更多的框，但是我决定不再继续分下去。

另一种优化调色板的技术被称为“octree quantization”，此技术由 Jeff Prosise 提出，并于 1996 年 8 月发表在《Microsoft Systems Journal》上（包含在 MSDN 的 CD 中）。

16.4.8 转换格式

DIBBLE 还允许将 DIB 从一种格式转换到另一种格式。这用到了 DIBCONV 文件中的 DibConvert 函数，如程序 16-25 所示。

程序 16-25　DIBCONV 文件

DIBCONV.H

```
/*-----------------------
    DIBCONV.H header file for DIBCONV.C
  -----------------------*/

HDIB DibConvert (HDIB hdibSrc, int iBitCountDst) ;
```

DIBCONV.C

```
/*-------------------------------
    DIBCONV.C——Converts DIBs from one format to another
               (c) Charles Petzold, 1998
  -------------------------------*/

#include <windows.h>
#include "dibhelp.h"
#include "dibpal.h"
#include "dibconv.h"

HDIB DibConvert (HDIB hdibSrc, int iBitCountDst)
{
     HDIB            hdibDst ;
     HPALETTE        hPalette ;
     int             i, x, y, cx, cy, iBitCountSrc, cColors ;
     PALETTEENTRY    pe ;
     RGBQUAD         rgb ;
     WORD            wNumEntries ;

     cx = DibWidth (hdibSrc) ;
     cy = DibHeight (hdibSrc) ;
     iBitCountSrc = DibBitCount (hdibSrc) ;

     if (iBitCountSrc == iBitCountDst)
```

```
    return NULL ;

    // DIB with color table to DIB with larger color table:

if ((iBitCountSrc < iBitCountDst) && (iBitCountDst <= 8))
{
     cColors = DibNumColors (hdibSrc) ;
     hdibDst = DibCreate (cx, cy, iBitCountDst, cColors) ;

     for (i = 0 ; i < cColors ; i++)
     {
          DibGetColor (hdibSrc, i, &rgb) ;
          DibSetColor (hdibDst, i, &rgb) ;
     }

     for (x = 0 ; x < cx ; x++)
     for (y = 0 ; y < cy ; y++)
     {
          DibSetPixel (hdibDst, x, y, DibGetPixel (hdibSrc, x, y)) ;
     }
}
    // Any DIB to DIB with no color table

else if (iBitCountDst >= 16)
{
     hdibDst = DibCreate (cx, cy, iBitCountDst, 0) ;

     for (x = 0 ; x < cx ; x++)
     for (y = 0 ; y < cy ; y++)
     {
          DibGetPixelColor (hdibSrc, x, y, &rgb) ;
          DibSetPixelColor (hdibDst, x, y, &rgb) ;
     }
}
    // DIB with no color table to 8-bit DIB

else if (iBitCountSrc >= 16 && iBitCountDst == 8)
{
     hPalette = DibPalMedianCut (hdibSrc, 6) ;
     GetObject (hPalette, sizeof (WORD), &wNumEntries) ;

     hdibDst = DibCreate (cx, cy, 8, wNumEntries) ;

     for (i = 0 ; i < (int) wNumEntries ; i++)
     {
          GetPaletteEntries (hPalette, i, 1, &pe) ;

          rgb.rgbRed   = pe.peRed ;
          rgb.rgbGreen = pe.peGreen ;
          rgb.rgbBlue  = pe.peBlue ;
          rgb.rgbReserved = 0 ;
```

```
            DibSetColor (hdibDst, i, &rgb) ;
        }

        for (x = 0 ; x < cx ; x++)
        for (y = 0 ; y < cy ; y++)
        {
            DibGetPixelColor (hdibSrc, x, y, &rgb) ;

            DibSetPixel (hdibDst, x, y,
                GetNearestPaletteIndex (hPalette,
                    RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
        }
        DeleteObject (hPalette) ;
    }
    // Any DIB to monochrome DIB

    else if (iBitCountDst == 1)
    {
        hdibDst = DibCreate (cx, cy, 1, 0) ;
        hPalette = DibPalUniformGrays (2) ;

        for (i = 0 ; i < 2 ; i++)
        {
            GetPaletteEntries (hPalette, i, 1, &pe) ;

            rgb.rgbRed   = pe.peRed ;
            rgb.rgbGreen = pe.peGreen ;
            rgb.rgbBlue  = pe.peBlue ;
            rgb.rgbReserved = 0 ;

            DibSetColor (hdibDst, i, &rgb) ;
        }

        for (x = 0 ; x < cx ; x++)
        for (y = 0 ; y < cy ; y++)
        {
            DibGetPixelColor (hdibSrc, x, y, &rgb) ;

            DibSetPixel (hdibDst, x, y,
                GetNearestPaletteIndex (hPalette,
                    RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
        }
        DeleteObject (hPalette) ;
    }
    // All non-monochrome DIBs to 4-bit DIB
    else if (iBitCountSrc >= 8 && iBitCountDst == 4)
    {
        hdibDst = DibCreate (cx, cy, 4, 0) ;
        hPalette = DibPalVga () ;
```

```
            for (i = 0 ; i < 16 ; i++)
            {
                GetPaletteEntries (hPalette, i, 1, &pe) ;

                rgb.rgbRed   = pe.peRed ;
                rgb.rgbGreen = pe.peGreen ;
                rgb.rgbBlue  = pe.peBlue ;
                rgb.rgbReserved = 0 ;

                DibSetColor (hdibDst, i, &rgb) ;
            }

            for (x = 0 ; x < cx ; x++)
            for (y = 0 ; y < cy ; y++)
            {
                DibGetPixelColor (hdibSrc, x, y, &rgb) ;

                DibSetPixel (hdibDst, x, y,
                    GetNearestPaletteIndex (hPalette,
                        RGB (rgb.rgbRed, rgb.rgbGreen, rgb.rgbBlue))) ;
            }
            DeleteObject (hPalette) ;
        }
        // Should not be necessary

    else
        hdibDst = NULL ;

    return hdibDst ;
}
```

将 DIB 从一种格式转换成另一种格式需要几种不同的方法。

要将带有颜色表的 DIB 转换成另一种也带有颜色表但有较大的像素宽度的 DIB（即将 1 位 DIB 转换成 4 位或 8 位 DIB，或将 4 位 DIB 转换成 8 位 DIB），所要做的只是通过调用 DibCreate 来创建新的 DIB，并在调用时带有希望的位数，以及与原始 DIB 中的颜色数相等的颜色数，然后函数复制像素位和颜色表条目。

如果新的 DIB 没有颜色表（即位数是 16、24 或 32），那么 DIB 只需要按新格式创建，而且通过调用 DibGetPixelColor 和 DibSetPixelColor 从现有的 DIB 中复制像素位。

下面的情况可能更普遍：现有的 DIB 没有颜色表（即位数是 16、24 或 32），而新的 DIB 每像素占 8 位。这种情况下，DibConvert 调用 DibPalMedianCut 来为图像创建优化的调色板。新 DIB 的颜色表设置为调色板中的 RGB 值。DibGetPixelColor 函数从现有的 DIB 中获得像素颜色。通过调用 GetNearestPaletteIndex 来转换成 8 位 DIB 中的像素值，并通过调用 DibSetPixel 将像素值保存到 DIB。

当 DIB 需要转换成单色 DIB 时，用包括两个条目——黑色和白色——的颜色表创建新的 DIB。另外，GetNearestPaletteIndex 有助于将现有 DIB 中的颜色转换成像素值 0 或

1。类似地，当 8 个像素位或更多位的 DIB 要转换成 4 位 DIB 时，可从 DibPalVga 函数获得 DIB 颜色表，同时 GetNearestPaletteIndex 也有助于计算像素值。

尽管 DIBBLE 显示了图像处理程序如何开始的基础，但是此程序始终没有彻底完成。显然还需要其他一些功能。遗憾的是，我们还有其他东西需要继续讨论。

第十七章

文本和字体

显示文本是本书所要解决的首要问题之一，现在我们来研究 Microsoft Windows 中各种有效字体和字号的使用方法，以及调整文本的方式。

Windows 3.1 引入的 TrueType 使程序员和用户以灵活的方式处理文本的能力得到了极大地增强。TrueType 是轮廓字体技术，由 Apple Computer 公司和 Microsoft 公司开发，并被许多字体制造商支持。由于 TrueType 字体能够无级缩放，并能应用于视频显示器和打印机，现在能够在 Windows 下实现真的 WYSIWYG(what you see is what you get：所见即所得)。TrueType 也便于制作“奇妙”字体，例如旋转的字母、内部填充图案的字母，或将它们用于剪辑区域，在本章我会演示它们。

17.1 简单的文本输出

让我们先来看看 Windows 用于文本输出的各种函数、影响文本的设备描述表属性，以及备用字体的用法。

17.1.1 文本输出函数

我已经在许多示例程序中使用过最常用的文本输出函数：

```
TextOut (hdc, xStart, yStart, pString, iCount) ;
```

参数 xStart 和 yStart 是逻辑坐标上字符串的起始点。通常，这是 Windows 开始绘制的第一个字母的左上角。TextOut 需要指向字符串的指针和字符串的长度，这个函数不能识别以 NULL 终止的字符串。

TextOut 函数的 xStart 和 yStart 参数的含义可由 SetTextAlign 函数改变。TA_LEFT、TA_RIGHT 和 TA_CENTER 标志影响使用 xStart 在水平方向上定位字符串的方式。默

认值是 TA_LEFT。如果在 SetTextAlign 函数中指定了 TA_RIGHT,则后面的 TextOut 调用会将字符串的最后一个字符定位于 xStart,如果指定了 TA_CENTER,则字符串的中心位于 xStart。

类似地,TA_TOP、TA_BOTTOM 和 TA_BASELINE 标志影响字符串的垂直位置。TA_TOP 是默认值,它意味着字符串的字母顶端位于 yStart,使用 TA_BOTTOM 意味着字符串位于 yStart 之上。可以使用 TA_BASELINE 定位字符串,使基线位于 yStart。基线是如小写字母 p、q、y 等字母下部的线。

如果使用 TA_UPDATECP 标志调用 SetTextAlign,Windows 就会忽略 TextOut 的 xStart 和 yStart 参数,而使用由 MoveToEx、LineTo 或更改当前位置的另一个函数设置的位置。TA_UPDATECP 标志也使 TextOut 函数将当前位置更新为字符串的结尾(TA_LEFT)或字符串的开头(TA_RIGHT)。这在使用多个 TextOut 调用显示一行文本时非常有用。当水平位置是 TA_CENTER 时,在 TextOut 调用后,当前位置不变。

您应该还记得,第四章的一系列 SYSTEMS 程序显示几列文本时,对每一列都需要调用一个 TextOut,其替代函数是 TabbedTextOut 函数:

```
TabbedTextOut (hdc, xStart, yStart, pString, iCount,
               iNumTabs, piTabStops, xTabOrigin) ;
```

如果文本字符串中含有嵌入的制表符('\t'或 0x09),则 TabbedTextOut 会根据传递给它的整数数组将制表符扩展为空格。

TabbedTextOut 的前 5 个参数与 TextOut 相同,第 6 个参数是制表位数,第 7 个是以像素为单位的制表位数组。例如,如果平均字符宽度是 8 个像素,而您希望每 5 个字符加一个制表位,则这个数组将包含 40、80、120,按递增顺序依此类推。

如果第 6 个和第 7 个参数是 0 或 NULL,则制表位按每 8 个平均字符宽度设置。如果第 6 个参数是 1,则第 7 个参数指向一个整数,表示制表位重复增大的倍数。(例如,如果第 6 个参数是 1,并且第 7 个参数指向值为 30 的变量,则制表位设置在 30、60、90……像素处。)最后一个参数给出了从制表位开始测量的逻辑 x 坐标,它与字符串的起始位置可能相同也可能不同。

另一个高级的文本输出函数是 ExtTextOut(前缀 Ext 表示它是扩展的):

```
ExtTextOut (hdc,xStart, yStart, iOptions, &rect,
            pString, iCount, pxDistance) ;
```

第 5 个参数是指向矩形结构的指针,在 iOptions 设置为 ETO_CLIPPED 时,该结构为剪裁矩形,在 iOptions 设置为 ETO_OPAQUE 时,该结构为用当前背景色填充的背景矩形。这两种选择您可以都采用,也可以都不采用。

最后一个参数是整数数组,它指定了字符串中连续字符的间隔。程序可以使用它

使字符间距变窄或变宽，因为有时需要在较窄的列中调整单个字。该参数可以设置为NULL来使用默认的字符间距。

用于写文本的高级函数是DrawText，我们第一次遇到它是在第三章讨论HELLOWIN程序时，它不指定坐标的起始位置，而是通过RECT结构类型定义希望显示文本的区域：

```
DrawText (hdc, pString, iCount, &rect, iFormat) ;
```

和其他文本输出函数一样，DrawText需要指向字符串的指针和字符串的长度。然而，如果在DrawText中使用以NULL结尾的字符串，就可以将iCount设置为－1，Windows会自动计算字符串的长度。

当iFormat设置为0时，Windows会将文本解释为一系列由回车符（'\r'或0x0D）或换行符（'\n'或0x0A）分隔的行。文本从矩形的左上角开始，回车符或换行符被理解为"新行"字符，因此Windows会结束当前行而开始新的一行。新的一行从矩形的左侧开始，在上一行的下面空开一个字符的高度（没有外部间隔）。包含字母的任何文本都应该显示在所剪裁矩形底部的右边或下边。

您可以使用iFormat参数更改DrawText的默认操作，iFormat由一个或多个标志组成。DT_LEFT标志（默认值）指定了左对齐的行，DT_RIGHT指定了右对齐的行，而DT_CENTER指定了位于矩形左边和右边中间的行。因为DT_LEFT的值是0，所以如果只需要左对齐，就不需要包含标识符。

如果您不希望将回车符或换行符解释为新行字符，则可以包括标识符DT_SINGLELINE。然后，Windows会把回车符和换行符解释为可显示的字符，而不是控制符。在使用DT_SINGLELINE时，还可以将行指定为位于矩形的顶端（DT_TOP）、底端（DT_BOTTOM），或者中间（DT_VCETER，V表示垂直）。

在显示多行文本时，Windows通常只在回车符或换行符处换行。然而，如果行的长度超出了矩形的宽度，则可以使用DT_WORDBREAK标志，它使Windows在行内字的末尾换行。对于单行或多行文本的显示，Windows会把超出矩形的文本部分截去，可以使用DT_NOCLIP跳过这个操作，这个标志还加快了函数的速度。当Windows确定多行文本的行距时，它通常使用不带外部间距的字符高度，如果您想在行距中加入外部间距，就可以使用标志DT_EXTERNALLEADING。

如果文本中包含制表符（'\t'或0x09），则需要包括标志DT_EXPANDTABS。在默认情况下，制表位设置于每八个字符的位置。通过使用标志DT_TABSTOP，您可以指定不同的制表设置，在这种情况下，iFormat的高位字节包含了每个新制表位的字符位置数值。不过我建议您避免使用DT_TABSTOP，因为iFormat的高位字节也用于其他标志。

DT_TABSTOP标志存在的问题，可以由新的函数DrawTextEx来解决，它含有一个额外的参数：

```
DrawTextEx (hdc, pString, iCount, &rect, iFormat, &drawtextparams) ;
```

最后一个参数是指向 DRAWTEXTPARAMS 结构的指针，它的定义如下：

```
typedef struct tagDRAWTEXTPARAMS
{
     UINT cbSize ;              // size of structure
     int iTabLength ;           // size of each tab stop
     int iLeftMargin ;          // left margin
     int iRightMargin ;         // right margin
     UINT uiLengthDrawn ;       // receives number of characters processed
} DRAWTEXTPARAMS, * LPDRAWTEXTPARAMS ;
```

中间的三个字段是以平均字符的增量为单位的。

17.1.2　文本的设备描述表属性

除了上面讨论的 SetTextAlign 外，其他几个设备描述表属性也对文本产生影响。在默认的设备描述表下，文本颜色是黑色，但您可以用下面的语句进行更改：

```
SetTextColor (hdc, rgbColor) ;
```

使用画笔的颜色和画刷的颜色，Windows 把 rgbColor 的值转换为纯色，您可以通过调用 GetTextColor 来获取当前文本的颜色。

Windows 在矩形的背景区域中显示文本，它可能根据背景模式的设置进行着色，也可能不这样做。您可以使用

```
SetBkMode (hdc, iMode) ;
```

更改背景模式，其中 iMode 的值为 OPAQUE 或 TRANSPARENT。默认的背景模式为 OPAQUE，它表示 Windows 使用背景颜色来填充矩形的背景。您可以使用

```
SetBkColor (hdc, rgbColor) ;
```

来改变背景颜色。rgbColor 的值是转换为纯色的值。默认背景色是白色。

如果两行文本靠得太近，其中一个的背景矩形就会遮盖另一个的文本。由于这种原因，我通常希望默认的背景模式是 TRANSPARENT。在背景模式为 TRANSPARENT 的情况下，Windows 会忽略背景色，也不对矩形背景区域着色。Windows 也使用背景模式和背景色对点和虚线之间的空隙及阴影刷中阴影间的区域着色，就像第五章讨论的那样。

许多 Windows 程序将 WHITE_BRUSH 指定为 Windows 用于擦出窗口背景的刷子，刷子在窗口类结构中指定。然而，您可能希望程序的窗口背景与用户在“控制面板”中设置的系统颜色保持一致，在这种情况下，可以在 WNDCLASS 结构中指定背景颜色的这种方式：

```
wndclass.hbrBackground = COLOR_WINDOW + 1 ;
```

当您想要在客户区书写文本时,可以使用当前系统颜色设置文本色和背景色:

```
SetTextColor (hdc, GetSysColor (COLOR_WINDOWTEXT)) ;
SetBkColor (hdc, GetSysColor (COLOR_WINDOW)) ;
```

完成这些以后,就可以使您的程序随系统颜色的变化而改变:

```
case WM_SYSCOLORCHANGE :
     InvalidateRect (hwnd, NULL, TRUE) ;
     break ;
```

另一个影响文本的设备描述表属性是字符间距。它的默认值是 0,表示 Windows 不在字符之间添加任何空间,但您可以使用以下函数插入空间:

```
SetTextCharacterExtra (hdc, iExtra) ;
```

参数 iExtra 是逻辑单位,Windows 将其转换为最接近的像素,它可以是 0。如果将 iExtra 取为负值(希望将字符紧紧压在一起),那么 Windows 会接受这个数值的绝对值,也就是说,您不能使 iExtra 的值小于 0。您可以通过调用 GetTextCharacterExtra 来获取当前的字符间距,Windows 在返回该值前会将像素间距转换为逻辑单位。

17.1.3 使用备用字体

当您调用 TextOut、TabbedTextOut、ExtTextOut、DrawText 或 DrawTextEx 书写文本时,Windows 使用设备描述表中当前选定的字体。字体定义了特定的字样和大小。以不同字体显示文本的最简单方法是使用 Windows 提供的备用字体,然而,它的范围是很有限的。

您可以调用下面的函数来获取某种备用字体的句柄:

```
hFont = GetStockObject (iFont) ;
```

其中,iFont 是几个标识符之一。然后,您就可以将该字体选入设备描述表:

```
SelectObject (hdc, hFont) ;
```

这些您也可以用一步完成:

```
SelectObject (hdc, GetStockObject (iFont)) ;
```

在默认的设备描述表中选定的字体称为系统字体,能够由 GetStockObject 的 SYSTEM_FONT 参数识别。这是非等宽的 ANSI 字符集字体。在 GetStockObject 中指定 SYSTEM_FIXED_FONT(我在本书的前面几个程序中应用过),可以获得等宽字体的句柄,这一字体与 Windows 3.0 以前的系统字体兼容。在希望所有的字体都具有相同宽度时,这是很方便的。

备用字体 OEM_FIXED_FONT 也称为终端字体，是 Windows 在 MS-DOS 命令提示窗口中使用的字体，它包括与原始 IBM-PC 扩展字符集兼容的字符集。Windows 在窗口标题栏、菜单和对话框的文本中使用 DEFULT_GUI_FONT。

当您将新字体选入设备描述表时，必须使用 GetTextMetrics 来计算字符的高度和平均宽度。如果选择了非等宽字体，那么一定要注意，字符的平均宽度只是个平均值，某些字符会比它宽或比它窄。在本章的后面，您会了解到确定由不同宽度字符组成的字符串总宽度的方法。

尽管 GetStockObject 确实提供了访问不同字体的最简单方式，但是您还不能充分控制 Windows 所提供的字体。不久，您会看到指定字体字样和大小的方法。

17.2　字体的背景

本章余下的部分致力于处理不同的字体。但是在您接触这些特定的代码前，对 Windows 使用字体的基本知识有一个深入的了解是很有好处的。

17.2.1　字体类型

Windows 支持两大类字体，即所谓的“GDI 字体”和“设备字体”。GDI 字体存储在硬盘的文件中，而设备字体是输出设备所固有的。例如，通常打印机都具有内置的设备字体集。

GDI 字体有三种类型：点阵字体、笔划字体和 TrueType 字体。

点阵字体有时也称为位图字体，因为每个字符都以位图像素图案的形式存储，每种点阵字体都有特定的纵横比和字符大小。Windows 通过简单地重复像素的行或列就可以由 GDI 点阵字体产生更大的字符。然而，只能以整数倍放大字体，并且不能超过一定的限度。由于这种原因，GDI 点阵字体又称为“不可缩放的”字体。它们不能随意地放大或缩小。点阵字体的主要优点是性能（显示速度很快）和可读性（因为是手工设计的，所以尽可能清晰）。

字体是通过字样名识别的，点阵字体的字样名为：

System	（用于 SYSTEM_FONT）
FixedSys	（用于 SYSTEM_FIXED_FONT）
Terminal	（用于 OEM_FIXED_FONT）
Courier	
MS Serif	
MS Sans Serif	（用于 DEFAULT_GUI_FONT）
Small Fonts	

每个点阵字体只有几种大小(不超过 6 种)。Courier 字体是定宽字体,外形与用打字机打出的字体相似。"Serif"指字体字母笔划在结束时拐个小弯。"sans serif"字体不是 serif 类的字体。在 Windows 的早期版本中,MS(Microsoft)Serif 和 MS Sans Serif 字体被称为 Tms Rmn(指它与 Times Roman 相似)和 Helv(与 Helvetica 相似)。Small Fonts 是专为显示小字设计的。

在 Windows 3.1 以前,除了 GDI 字体外,Windows 所提供的字体只有笔划字体。笔划字体是以"连接点"的方式定义的一系列线段,笔划字体可以连续地缩放,这意味着同样的字体可以用于具有任何分辨率的图形输出设备,并且字体可以放大或缩小到任意尺寸。不过,它的性能不好,小字体的可读性也很糟,而大字体由于笔划是单根直线而显得很单薄。笔划字体有时也称为绘图仪字体,因为它们特别适合于绘图仪,但是不适合于别的场合。笔划字体的字样有:Modern、Roman 和 Script。

对于 GDI 点阵字体和 GDI 笔划字体,Windows 都可以"合成"粗体、斜体、加下划线和加删除线,而不需要为每种属性另外存储字体。例如,对于斜体,Windows 只要将字符的上部向右移动就可以了。

接下来是 TrueType,我在本章的余下部分主要讨论它。

17.2.2 TrueType 字体

TrueType 字体的单个字符是通过填充的直线和曲线的轮廓来定义的。Windows 可以通过改变定义轮廓的坐标对 TrueType 字体进行缩放。

当程序开始使用特定大小的 TrueType 字体时,Windows"点阵化"字体。这就是说 Windows 使用 TrueType 字体文件中包括的"提示"对每个字符的连接直线和曲线的坐标进行缩放。这些提示可以补偿误差,避免合成的字符变得很难看。(例如,在某些字体中,大写 H 的两竖应该一样宽,但盲目地缩放字体可能会导致其中一竖的像素比另一竖宽。有了提示就可以避免这些现象发生。)然后,每个字符的合成轮廓用于创建字符的位图,这些位图储存在内存以备将来使用。

最初,Windows 使用了 13 种 TrueType 字体,它们的字样名如下:

Courier New
Courier New Bold
Courier New Italic
Courier New Bold Italic
Times New Roman
Times New Roman Bold
Times New Roman Italic
Times New Roman Bold Italic

Arial
Arial Bold
Arial Italic
Arial Bold Italic
Symbol

在新的 Windows 版本中，这个列表更长了。特别指出，我将使用 Lucida Sans Unicode 字体，它包括了一些在世界其他地方使用的字母表。

三个主要字体族与点阵字体相似，Courier New 是等宽字体。它看起来就像是打字机输出的字体。Times New Roman 是 Times 字体的复制品，该字体最初为 Times of London 设计，并用在许多印刷材料上，它具有很好的可读性。Arial 是 Helvetica 字体的复制品，是一种 sans serif 字体。Symbol 字体包含了手写符号集。

17.2.3　属性或风格

在上面的 TrueType 字体列表中，您会注意到，Courier、Times New Roman 和 Arial 的粗体和斜体是带有自己字样名的单独字体，这一命名与传统的版式一致。然而，计算机用户认为粗体和斜体只是已有字体的特殊“属性”。Windows 在定义点阵字体命名、列举和选择的方式时，采用了属性的方法。但对于 TrueType 字体，更倾向于使用传统的命名方式。

这种冲突在 Windows 中还没有完全解决，简而言之，您可以完全通过命名或特定属性来选择字体。然而在处理字体列举时，应用程序需要系统中的字体列表。正如您所预料的那样，这种双重处理使问题复杂化了。

17.2.4　磅值

在传统的版式中，您可以用字样名和大小来指定字体，字体的大小以磅为单位来表示。1 磅约为 1/72 英寸——事实上它们很接近，以至于在计算机中它通常被定义为 1/72 英寸。磅值通常描述为字母顶端（不包括发音符号）到字母底端的高度，例如字母“bq”的总高度。这是一个考虑字体大小的简单方式，但它通常不是很精确。

字体的磅值实际上是排版设计的概念，而不是度量概念。特定字体中字符的大小可能会大于或小于其磅值所表示的大小。在传统的排版中，使用磅值来指定字体的大小，在计算机排版中，还有其他方法来确定字符的实际大小。

17.2.5　间隔和间距

在第四章曾提到，可以通过调用 GetTextMetrics 来获取设备描述表中当前选定的字体信息，我们也多次使用过这个函数。图 4-3 显示了 FONTMETRIC 结构中字体的垂直大小。

TEXTMETRIC 结构的另一个字段是 tmExternalLeading,“间隔(leading)”来自排字工人在金属字块间插入的铅条,它用以在两行文本之间产生空白。tmInternalLeading 值与为发音符号保留的空间有关,tmExternalLeading 表示字符的连续行之间留的附加空间。程序员可以使用或忽略外部的间隔值。

当我们说一个字体是 8 磅或 12 磅时,指的是不带内部间隔的高度。某种大写字母上的发音符号占据了分隔行的间距。这样,TEXTMETRIC 结构的 tmHeight 值实际指行间距而不是字体的磅值。字体的磅值可由 tmHeight 减 tmInternalLeading 得到。

17.2.6 逻辑英寸问题

正如我们在 5.2.4 节中所讨论的,Windows 98 将系统字体定义为具有 12 磅行距的 10 磅字体。根据在“显示属性”对话框中选择的是“小字体”还是“大字体”,该字体的 tmHeight 值为 16 或 20 像素,tmHeight 减去 tmInternalLeading 的值为 13 或 16 像素。这样,字体的选择就暗指以每英寸的点数为单位的设备分辨率,选择“小字体”即为 96 dpi,选择“大字体”即为 120 dpi。

您可以用 LOGPIXELSX 或 LOGPIXELSY 参数调用 GetDeviceCaps 来获取设备分辨率。因此,96 或 120 像素在屏幕上占有的度量距离又称为“逻辑英寸”。如果您用尺子测量屏幕并计算像素,就可能发现逻辑英寸要比实际英寸大一些,为什么会这样呢?

在纸张上,每英寸放置 14 个 8 磅的字符很便于阅读。如果在编写字处理或写作应用程序时,可能希望在显示器上显示清晰的 8 磅字体,但如果使用视频显示器的实际尺寸,就没有足够的像素来清晰地显示字符。即使显示器具有足够的分辨率,在屏幕上阅读 8 磅字体仍然会有问题。当人们阅读纸上的印刷物时,眼睛与文本的距离通常为 1 英尺,而使用视频显示器时,这个距离通常为 2 英尺。

逻辑英寸有效地对屏幕进行了放大,可显示小至 8 磅的清晰字体。而且每英寸 96 点使 640 像素的最小显示大小约为 6.5 英寸。这恰恰是在页边距为 1 英寸的8.5英寸宽的纸上打印的文本的宽度。因而逻辑英寸也利用了屏幕宽度,尽可能大地显示文本。

您可能还记得在第五章中,Windows NT 的做法有些不同。在 Windows NT 中,从 GetDeviceCaps 中得到的 LOGPIXELSX 值(每英寸的像素数)$\neq \frac{\text{HORZRES 值(像素数)}}{\text{HORZSIZE 值(毫米数)}} \times 25.4$。与此类似,LOGPIXELSY、VERTRES 和 VERTSIZE 也不一致。Windows 在为不同映射方式计算窗口和偏移范围时,使用 HORZRES、HORZSIZE、VERTRES 和 VERTSIZE 值;然而,对文本进行显示的程序最好不要使用基于 LOGPIXELSX 和 LOGPIXELSY 假定的显示分辨率,这一点与 Windows 98 更为一致。

所以,在 Windows NT 下,当程序以特定的磅值显示文本时,它可能不使用 Windows 提供的映射方式,程序根据与 Windows 98 一致的每英寸的逻辑像素数来定义自己的映射方式。我将这种用于文本的映射方式称为“Logical Twips”映射方式。可以设置如下:

```
SetMapMode (hdc, MM_ANISOTROPIC) ;
SetWindowExtEx (hdc, 1440, 1440, NULL) ;
SetViewportExt (hdc, GetDeviceCaps (hdc, LOGPIXELSX),
                     GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;
```

使用这种映射方式设置，能够以磅值的 20 倍来指定字体大小，例如，为 12 磅字取 240。注意，与 MM_TWIPS 映射方式不同，y 值在屏幕中向下增长，这在显示文本的连续行时很方便。

请记住，逻辑英寸与实际英寸间的差异仅对显示器存在。在打印设备上，GDI 和尺子是完全一致的。

17.3　逻辑字体

既然我们已经明确了逻辑英寸和逻辑单位的概念，现在我们就来讨论逻辑字体。

逻辑字体是一个 GDI 对象，它的句柄存储在 HFONT 类型的变量中，逻辑字体是字体的描述。和逻辑画笔及逻辑刷子一样，它是抽象的对象，只有当应用程序调用 SelectObject 将它选入设备描述表时，它才成为真实的对象。例如，对于逻辑画笔，您可以为画笔指定任意的颜色，但是在您将画笔选入设备描述表时，Windows 才将其转换为设备中有效的颜色。只有这时，Windows 才知道设备的色彩能力。

17.3.1　逻辑字体的创建和选择

可以通过调用 CreateFont 或 CreateFontIndirect 来创建逻辑字体。CreateFontIndirect 函数接受一个指向 LOGFONT 结构的指针，该结构有 14 个字段。CreateFont 函数接受 14 个参数，它们与 LOGFONT 结构的 14 个字段形式相同。它们是仅有的两个创建逻辑字体的函数。（我提到这一点，是因为 Windows 中有许多用于其他字体操作的函数。）因为很难记住 14 个字段，所以很少使用 CreateFont，因此，我主要讨论 CreateFontIndirect。

有三种基本的方式用于定义 LOGFONT 结构中的字段，以便调用 CreateFontIndirect：

- 您可以简单地将 LOGFONT 结构的字段设置为所需的字体特征。在这种情况下，在调用 SelectObject 时，Windows 使用“字体映射”算法从设备上有效的字体中选择与这些特征最匹配的字体。由于这依赖于视频显示器和打印机上的有效字体，所以其结果可能与您的要求有相当大的差别。
- 您可以列举设备上的所有字体并从中选择，甚至用对话框把它们显示给用户。我将在本章后面讨论字体列举函数。不过，它们现在已经不常用了，因为第三种方法也可以进行列举。
- 您可以采用简单的方法并调用 ChooseFont 函数，我在第十一章曾讨论过这个函数，能够使用 LOGFONT 结构直接创建字体。

在本章中，我使用第一种和第三种方法。

下面是创建、选择和删除逻辑字体的过程：

1. 通过调用 CreateFont 或 CreateFontIndirect 来创建逻辑字体，这些函数返回 HFONT 类型的逻辑字体句柄。
2. 使用 SelectObject 将逻辑字体选入设备描述表，Windows 会选择与逻辑字体最匹配的真实字体。
3. 使用 GetTextMetrics（及可能用到的其他函数）确定真实字体的大小和特征。在该字体选入设备描述表后，可以使用这些信息适当地设置文本的间距。
4. 在结束使用逻辑字体后，调用 DeleteObject 删除逻辑字体，当字体选入有效的设备描述表时，不要删除字体，也不要删除备用字体。

GetTextFace 函数使程序能够确定当前选入设备描述表的字体名：

```
GetTextFace (hdc, sizeof (szFaceName) / sizeof (TCHAR), szFaceName) ;
```

详细的字体信息可以从 GetTextMetrics 中得到：

```
GetTextMetrics (hdc, &textmetric) ;
```

其中，textmetric 是 TEXTMETRIC 类型的变量，它具有 20 个字段。

稍后我将详细讨论 LOGFONT 和 TEXTMETRIC 结构的字段，这两个结构有一些相似的字段，所以它们容易混淆。现在您只需记住，LOGFONT 用于定义逻辑字体，而 TEXTMETRIC 用于获取当前选入设备描述表中的字体信息。

17.3.2 PICKFONT 程序

使用程序 17-1 所示的 PICKFONT，可以定义 LOGFONT 结构的许多字段。这个程序创建逻辑字体，并在逻辑字体选入设备描述表后显示真实字体的特征。这是个方便的程序，通过它我们可以了解逻辑字体映射为真实字体的方式。

程序 17-1　PICKFONT 程序

PICKFONT.C

```
/*----------------------
   PICKFONT.C——Create Logical Font
              (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>
#include "resource.h"

   // Structure shared between main window and dialog box
```

```
typedef struct
{
     int          iDevice, iMapMode ;
     BOOL         fMatchAspect ;
     BOOL         fAdvGraphics ;
     LOGFONT      lf ;
     TEXTMETRIC   tm ;
     TCHAR        szFaceName [LF_FULLFACESIZE] ;
}
DLGPARAMS ;

     // Formatting for BCHAR fields of TEXTMETRIC structure

# ifdef UNICODE
# define BCHARFORM TEXT ("0x%04X")
# else
# define BCHARFORM TEXT ("0x%02X")
# endif

     // Global variables

HWND    hdlg ;

TCHAR szAppName[ ] = TEXT ("PickFont") ;

     // Forward declarations of functions

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
BOOL    CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;
void SetLogFontFromFields   (HWND hdlg, DLGPARAMS * pdp) ;
void SetFieldsFromTextMetric (HWND hdlg, DLGPARAMS * pdp) ;
void MySetMapMode (HDC hdc, int iMapMode) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
```

```
     wndclass.lpszMenuName  = szAppName ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
               szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("PickFont: Create Logical Font"),
                          WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          if (hdlg == 0 || !IsDialogMessage (hdlg, &msg))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static DLGPARAMS dp ;
     static TCHAR     szText[] = TEXT (" \x41 \x42 \x43 \x44 \x45 ")
                                 TEXT (" \x61 \x62 \x63 \x64 \x65 ")

                                 TEXT (" \xC0 \xC1 \xC2 \xC3 \xC4 \xC5 ")
                                 TEXT (" \xE0 \xE1 \xE2 \xE3 \xE4 \xE5 ")
# ifdef UNICODE
                                 TEXT (" \x0390 \x0391 \x0392 \x0393 \x0394 \x0395 ")
                                 TEXT (" \x03B0 \x03B1 \x03B2 \x03B3 \x03B4 \x03B5 ")

                                 TEXT (" \x0410 \x0411 \x0412 \x0413 \x0414 \x0415 ")
                                 TEXT (" \x0430 \x0431 \x0432 \x0433 \x0434 \x0435 ")

                                 TEXT (" \x5000 \x5001 \x5002 \x5003 \x5004")
# endif
                                 ;
```

```
HDC             hdc ;
PAINTSTRUCT     ps ;
RECT            rect ;

switch (message)
{
case WM_CREATE:
    dp.iDevice = IDM_DEVICE_SCREEN ;

    hdlg = CreateDialogParam (((LPCREATESTRUCT) lParam) -> hInstance,
                              szAppName, hwnd, DlgProc, (LPARAM) &dp) ;
    return 0 ;

case WM_SETFOCUS:
    SetFocus (hdlg) ;
    return 0 ;

case WM_COMMAND:
    switch (LOWORD (wParam))
    {
    case IDM_DEVICE_SCREEN:
    case IDM_DEVICE_PRINTER:
        CheckMenuItem (GetMenu (hwnd), dp.iDevice, MF_UNCHECKED) ;
        dp.iDevice = LOWORD (wParam) ;
        CheckMenuItem (GetMenu (hwnd), dp.iDevice, MF_CHECKED) ;
        SendMessage (hwnd, WM_COMMAND, IDOK, 0) ;
        return 0 ;
    }
    break ;

case WM_PAINT:
    hdc = BeginPaint (hwnd, &ps) ;

        // Set graphics mode so escapement works in Windows NT

    SetGraphicsMode (hdc, dp.fAdvGraphics ? GM_ADVANCED : GM_COMPATIBLE) ;

        // Set the mapping mode and the mapper flag

    MySetMapMode (hdc, dp.iMapMode) ;
    SetMapperFlags (hdc, dp.fMatchAspect) ;

        // Find the point to begin drawing text

    GetClientRect (hdlg, &rect) ;
    rect.bottom += 1 ;
    DPtoLP (hdc, (PPOINT) &rect, 2) ;
```

```
            // Create and select the font; display the text

            SelectObject (hdc, CreateFontIndirect (&dp.lf)) ;
            TextOut (hdc, rect.left, rect.bottom, szText, lstrlen (szText)) ;

            DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK DlgProc (HWND hdlg, UINT message, WPARAM wParam, LPARAM lParam)
{
        static DLGPARAMS * pdp ;
        static PRINTDLG    pd = { sizeof (PRINTDLG) } ;
        HDC                hdcDevice ;
        HFONT              hFont ;

        switch (message)
        {
        case WM_INITDIALOG:
                // Save pointer to dialog-parameters structure in WndProc

            pdp = (DLGPARAMS *) lParam ;

            SendDlgItemMessage (hdlg, IDC_LF_FACENAME, EM_LIMITTEXT,
                                LF_FACESIZE - 1, 0) ;

            CheckRadioButton (hdlg, IDC_OUT_DEFAULT, IDC_OUT_OUTLINE,
                              IDC_OUT_DEFAULT) ;

            CheckRadioButton (hdlg, IDC_DEFAULT_QUALITY, IDC_PROOF_QUALITY,
                              IDC_DEFAULT_QUALITY) ;

            CheckRadioButton (hdlg, IDC_DEFAULT_PITCH, IDC_VARIABLE_PITCH,
                              IDC_DEFAULT_PITCH) ;

            CheckRadioButton (hdlg, IDC_FF_DONTCARE, IDC_FF_DECORATIVE,
                              IDC_FF_DONTCARE) ;

            CheckRadioButton (hdlg, IDC_MM_TEXT, IDC_MM_LOGTWIPS,
```

```
                                IDC_MM_TEXT) ;

        SendMessage (hdlg, WM_COMMAND, IDOK, 0) ;
                                // fall through
    case WM_SETFOCUS:
        SetFocus (GetDlgItem (hdlg, IDC_LF_HEIGHT)) ;
        return FALSE ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDC_CHARSET_HELP:
            MessageBox (hdlg,
                        TEXT ("0 = Ansi \ n")
                        TEXT ("1 = Default \ n")
                        TEXT ("2 = Symbol \ n")
                        TEXT ("128 = Shift JIS (Japanese) \ n")
                        TEXT ("129 = Hangul (Korean) \ n")
                        TEXT ("130 = Johab (Korean) \ n")
                        TEXT ("134 = GB 2312 (Simplified Chinese) \ n")
                        TEXT ("136 = Chinese Big 5 (Traditional Chinese) \ n")
                        TEXT ("177 = Hebrew \ n")
                        TEXT ("178 = Arabic \ n")
                        TEXT ("161 = Greek \ n")
                        TEXT ("162 = Turkish \ n")
                        TEXT ("163 = Vietnamese \ n")
                        TEXT ("204 = Russian \ n")
                        TEXT ("222 = Thai \ n")
                        TEXT ("238 = East European \ n")
                        TEXT ("255 = OEM"),
                        szAppName, MB_OK | MB_ICONINFORMATION) ;
            return TRUE ;

            // These radio buttons set the lfOutPrecision field

        case IDC_OUT_DEFAULT:
            pdp -> lf.lfOutPrecision = OUT_DEFAULT_PRECIS ;
            return TRUE ;

        case IDC_OUT_STRING:
            pdp -> lf.lfOutPrecision = OUT_STRING_PRECIS ;
            return TRUE ;

        case IDC_OUT_CHARACTER:
            pdp -> lf.lfOutPrecision = OUT_CHARACTER_PRECIS ;
            return TRUE ;
```

```
case IDC_OUT_STROKE:
    pdp -> lf.lfOutPrecision = OUT_STROKE_PRECIS ;
    return TRUE ;

case IDC_OUT_TT:
    pdp -> lf.lfOutPrecision = OUT_TT_PRECIS ;
    return TRUE ;

case IDC_OUT_DEVICE:
    pdp -> lf.lfOutPrecision = OUT_DEVICE_PRECIS ;
    return TRUE ;

case IDC_OUT_RASTER:
    pdp -> lf.lfOutPrecision = OUT_RASTER_PRECIS ;
    return TRUE ;

case IDC_OUT_TT_ONLY:
    pdp -> lf.lfOutPrecision = OUT_TT_ONLY_PRECIS ;
    return TRUE ;

case IDC_OUT_OUTLINE:
    pdp -> lf.lfOutPrecision = OUT_OUTLINE_PRECIS ;
    return TRUE ;

    // These three radio buttons set the lfQuality field

case IDC_DEFAULT_QUALITY:
    pdp -> lf.lfQuality = DEFAULT_QUALITY ;
    return TRUE ;

case IDC_DRAFT_QUALITY:
    pdp -> lf.lfQuality = DRAFT_QUALITY ;
    return TRUE ;

case IDC_PROOF_QUALITY:
    pdp -> lf.lfQuality = PROOF_QUALITY ;
    return TRUE ;

    // These three radio buttons set the lower nibble
    // of the lfPitchAndFamily field

case IDC_DEFAULT_PITCH:
    pdp -> lf.lfPitchAndFamily =
          (0xF0 & pdp -> lf.lfPitchAndFamily) | DEFAULT_PITCH ;
    return TRUE ;
```

```
case IDC_FIXED_PITCH:
    pdp -> lf.lfPitchAndFamily =
          (0xF0 & pdp -> lf.lfPitchAndFamily) | FIXED_PITCH ;
    return TRUE ;

case IDC_VARIABLE_PITCH:
    pdp -> lf.lfPitchAndFamily =
          (0xF0 & pdp -> lf.lfPitchAndFamily) | VARIABLE_PITCH ;
    return TRUE ;

    // These six radio buttons set the upper nibble
    //    of the lfPitchAndFamily field

case IDC_FF_DONTCARE:
    pdp -> lf.lfPitchAndFamily =
          (0x0F & pdp -> lf.lfPitchAndFamily) | FF_DONTCARE ;
    return TRUE ;

case IDC_FF_ROMAN:
    pdp -> lf.lfPitchAndFamily =
          (0x0F & pdp -> lf.lfPitchAndFamily) | FF_ROMAN ;
    return TRUE ;

case IDC_FF_SWISS:
    pdp -> lf.lfPitchAndFamily =
          (0x0F & pdp -> lf.lfPitchAndFamily) | FF_SWISS ;
    return TRUE ;

case IDC_FF_MODERN:
    pdp -> lf.lfPitchAndFamily =
          (0x0F & pdp -> lf.lfPitchAndFamily) | FF_MODERN ;
    return TRUE ;

case IDC_FF_SCRIPT:
    pdp -> lf.lfPitchAndFamily =
          (0x0F & pdp -> lf.lfPitchAndFamily) | FF_SCRIPT ;
    return TRUE ;

case IDC_FF_DECORATIVE:
    pdp -> lf.lfPitchAndFamily =
          (0x0F & pdp -> lf.lfPitchAndFamily) | FF_DECORATIVE ;
    return TRUE ;

    // Mapping mode:

case IDC_MM_TEXT:
```

```
case IDC_MM_LOMETRIC:
case IDC_MM_HIMETRIC:
case IDC_MM_LOENGLISH:
case IDC_MM_HIENGLISH:
case IDC_MM_TWIPS:
case IDC_MM_LOGTWIPS:
     pdp -> iMapMode = LOWORD (wParam) ;
     return TRUE ;

     // OK button pressed
     // - - - - - - - - - - - - - - - - -

case IDOK:
          // Get LOGFONT structure

     SetLogFontFromFields (hdlg, pdp) ;

          // Set Match-Aspect and Advanced Graphics flags

     pdp -> fMatchAspect = IsDlgButtonChecked (hdlg, IDC_MATCH_ASPECT) ;
     pdp -> fAdvGraphics = IsDlgButtonChecked (hdlg, IDC_ADV_GRAPHICS) ;

          // Get Information Context
     if (pdp -> iDevice == IDM_DEVICE_SCREEN)
     {
          hdcDevice = CreateIC (TEXT ("DISPLAY"), NULL, NULL, NULL) ;
     }
     else
     {
          pd.hwndOwner = hdlg ;
          pd.Flags = PD_RETURNDEFAULT | PD_RETURNIC ;
          pd.hDevNames = NULL ;
          pd.hDevMode = NULL ;

          PrintDlg (&pd) ;

          hdcDevice = pd.hDC ;
     }
          // Set the mapping mode and the mapper flag

     MySetMapMode (hdcDevice, pdp -> iMapMode) ;
     SetMapperFlags (hdcDevice, pdp -> fMatchAspect) ;

          // Create font and select it into IC

     hFont = CreateFontIndirect (&pdp -> lf) ;
     SelectObject (hdcDevice, hFont) ;
```

```
                    // Get the text metrics and face name

                GetTextMetrics (hdcDevice, &pdp -> tm) ;
                GetTextFace (hdcDevice, LF_FULLFACESIZE, pdp -> szFaceName) ;
                DeleteDC (hdcDevice) ;
                DeleteObject (hFont) ;

                    // Update dialog fields and invalidate main window

                SetFieldsFromTextMetric (hdlg, pdp) ;
                InvalidateRect (GetParent (hdlg), NULL, TRUE) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}

void SetLogFontFromFields (HWND hdlg, DLGPARAMS * pdp)
{
        pdp -> lf.lfHeight          = GetDlgItemInt (hdlg, IDC_LF_HEIGHT,     NULL, TRUE) ;
        pdp -> lf.lfWidth           = GetDlgItemInt (hdlg, IDC_LF_WIDTH,      NULL, TRUE) ;
        pdp -> lf.lfEscapement      = GetDlgItemInt (hdlg, IDC_LF_ESCAPE,     NULL, TRUE) ;
        pdp -> lf.lfOrientation     = GetDlgItemInt (hdlg, IDC_LF_ORIENT,     NULL, TRUE) ;
        pdp -> lf.lfWeight          = GetDlgItemInt (hdlg, IDC_LF_WEIGHT,     NULL, TRUE) ;
        pdp -> lf.lfCharSet         = GetDlgItemInt (hdlg, IDC_LF_CHARSET,    NULL, FALSE) ;

        pdp -> lf.lfItalic =
                    IsDlgButtonChecked (hdlg, IDC_LF_ITALIC) == BST_CHECKED ;
        pdp -> lf.lfUnderline =
                    IsDlgButtonChecked (hdlg, IDC_LF_UNDER) == BST_CHECKED ;
        pdp -> lf.lfStrikeOut =
                    IsDlgButtonChecked (hdlg, IDC_LF_STRIKE) == BST_CHECKED ;
        GetDlgItemText (hdlg, IDC_LF_FACENAME, pdp -> lf.lfFaceName, LF_FACESIZE) ;
}

void SetFieldsFromTextMetric (HWND hdlg, DLGPARAMS * pdp)
{
        TCHAR    szBuffer [10] ;
        TCHAR * szYes = TEXT ("Yes") ;
        TCHAR * szNo = TEXT ("No") ;
        TCHAR * szFamily [] = { TEXT ("Don't Know"), TEXT ("Roman"),
                                TEXT ("Swiss"),      TEXT ("Modern"),
                                TEXT ("Script"),     TEXT ("Decorative"),
                                TEXT ("Undefined") } ;

        SetDlgItemInt (hdlg, IDC_TM_HEIGHT,     pdp -> tm.tmHeight,                TRUE) ;
```

```
     SetDlgItemInt (hdlg, IDC_TM_ASCENT,     pdp -> tm.tmAscent,          TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_DESCENT,    pdp -> tm.tmDescent,         TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_INTLEAD,    pdp -> tm.tmInternalLeading, TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_EXTLEAD,    pdp -> tm.tmExternalLeading, TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_AVECHAR,    pdp -> tm.tmAveCharWidth,    TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_MAXCHAR,    pdp -> tm.tmMaxCharWidth,    TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_WEIGHT,     pdp -> tm.tmWeight,          TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_OVERHANG,   pdp -> tm.tmOverhang,       TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_DIGASPX,    pdp -> tm.tmDigitizedAspectX, TRUE) ;
     SetDlgItemInt (hdlg, IDC_TM_DIGASPY,    pdp -> tm.tmDigitizedAspectY, TRUE) ;

     wsprintf (szBuffer, BCHARFORM, pdp -> tm.tmFirstChar) ;
     SetDlgItemText (hdlg, IDC_TM_FIRSTCHAR, szBuffer) ;

     wsprintf (szBuffer, BCHARFORM, pdp -> tm.tmLastChar) ;
     SetDlgItemText (hdlg, IDC_TM_LASTCHAR, szBuffer) ;

     wsprintf (szBuffer, BCHARFORM, pdp -> tm.tmDefaultChar) ;
     SetDlgItemText (hdlg, IDC_TM_DEFCHAR, szBuffer) ;

     wsprintf (szBuffer, BCHARFORM, pdp -> tm.tmBreakChar) ;
     SetDlgItemText (hdlg, IDC_TM_BREAKCHAR, szBuffer) ;

     SetDlgItemText (hdlg, IDC_TM_ITALIC, pdp -> tm.tmItalic       ? szYes : szNo) ;
     SetDlgItemText (hdlg, IDC_TM_UNDER, pdp -> tm.tmUnderlined    ? szYes : szNo) ;
     SetDlgItemText (hdlg, IDC_TM_STRUCK, pdp -> tm.tmStruckOut    ? szYes : szNo) ;

     SetDlgItemText (hdlg, IDC_TM_VARIABLE,
          TMPF_FIXED_PITCH & pdp -> tm.tmPitchAndFamily ? szYes : szNo) ;

     SetDlgItemText (hdlg, IDC_TM_VECTOR,
          TMPF_VECTOR & pdp -> tm.tmPitchAndFamily ? szYes : szNo) ;

     SetDlgItemText (hdlg, IDC_TM_TRUETYPE,
          TMPF_TRUETYPE & pdp -> tm.tmPitchAndFamily ? szYes : szNo) ;

     SetDlgItemText (hdlg, IDC_TM_DEVICE,
          TMPF_DEVICE & pdp -> tm.tmPitchAndFamily ? szYes : szNo) ;

     SetDlgItemText (hdlg, IDC_TM_FAMILY,
          szFamily [min (6, pdp -> tm.tmPitchAndFamily >> 4)]) ;

     SetDlgItemInt  (hdlg, IDC_TM_CHARSET,   pdp -> tm.tmCharSet, FALSE) ;
     SetDlgItemText (hdlg, IDC_TM_FACENAME, pdp -> szFaceName) ;
}
```

```
void MySetMapMode (HDC hdc, int iMapMode)
{
     switch (iMapMode)
     {
     case IDC_MM_TEXT:          SetMapMode (hdc, MM_TEXT) ;          break ;
     case IDC_MM_LOMETRIC:      SetMapMode (hdc, MM_LOMETRIC) ;      break ;
     case IDC_MM_HIMETRIC:      SetMapMode (hdc, MM_HIMETRIC) ;      break ;
     case IDC_MM_LOENGLISH:     SetMapMode (hdc, MM_LOENGLISH) ;     break ;
     case IDC_MM_HIENGLISH:     SetMapMode (hdc, MM_HIENGLISH) ;     break ;
     case IDC_MM_TWIPS:         SetMapMode (hdc, MM_TWIPS) ;         break ;
     case IDC_MM_LOGTWIPS:
          SetMapMode (hdc, MM_ANISOTROPIC) ;
          SetWindowExtEx (hdc, 1440, 1440, NULL) ;
          SetViewportExtEx (hdc, GetDeviceCaps (hdc, LOGPIXELSX),
                                 GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;
          break ;
     }
}
```

PICKFONT.RC

```
// Microsoft Developer Studio generated resource script.
#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog

PICKFONT DIALOG DISCARDABLE 0, 0, 348, 308
STYLE WS_CHILD | WS_VISIBLE | WS_BORDER
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "&Height:",IDC_STATIC,8,10,44,8
    EDITTEXT        IDC_LF_HEIGHT,64,8,24,12,ES_AUTOHSCROLL
    LTEXT           "&Width",IDC_STATIC,8,26,44,8
    EDITTEXT        IDC_LF_WIDTH,64,24,24,12,ES_AUTOHSCROLL
    LTEXT           "Escapement:",IDC_STATIC,8,42,44,8
    EDITTEXT        IDC_LF_ESCAPE,64,40,24,12,ES_AUTOHSCROLL
    LTEXT           "Orientation:",IDC_STATIC,8,58,44,8
    EDITTEXT        IDC_LF_ORIENT,64,56,24,12,ES_AUTOHSCROLL
    LTEXT           "Weight:",IDC_STATIC,8,74,44,8
    EDITTEXT        IDC_LF_WEIGHT,64,74,24,12,ES_AUTOHSCROLL
    GROUPBOX        "Mapping Mode",IDC_STATIC,97,3,96,90,WS_GROUP
    CONTROL         "Text",IDC_MM_TEXT,"Button",BS_AUTORADIOBUTTON,104,13,56,
                    8
```

```
CONTROL         "Low Metric", IDC_MM_LOMETRIC, "Button", BS_AUTORADIOBUTTON,
                104,24,56,8
CONTROL         "High Metric", IDC_MM_HIMETRIC, "Button",
                BS_AUTORADIOBUTTON, 104,35,56,8
CONTROL         "Low English", IDC_MM_LOENGLISH, "Button",
                BS_AUTORADIOBUTTON, 104,46,56,8
CONTROL         "High English", IDC_MM_HIENGLISH, "Button",
                BS_AUTORADIOBUTTON, 104,57,56,8
CONTROL         "Twips", IDC_MM_TWIPS, "Button", BS_AUTORADIOBUTTON, 104,68,
                56,8
CONTROL         "Logical Twips", IDC_MM_LOGTWIPS, "Button",
                BS_AUTORADIOBUTTON, 104,79,64,8
CONTROL         "Italic", IDC_LF_ITALIC, "Button", BS_AUTOCHECKBOX |
                WS_TABSTOP, 8,90,48,12
CONTROL         "Underline", IDC_LF_UNDER, "Button", BS_AUTOCHECKBOX |
                WS_TABSTOP, 8,104,48,12
CONTROL         "Strike Out", IDC_LF_STRIKE, "Button", BS_AUTOCHECKBOX |
                WS_TABSTOP, 8,118,48,12
CONTROL         "Match Aspect", IDC_MATCH_ASPECT, "Button", BS_AUTOCHECKBOX |
                WS_TABSTOP, 60,104,62,8
CONTROL         "Adv Grfx Mode", IDC_ADV_GRAPHICS, "Button",
                BS_AUTOCHECKBOX | WS_TABSTOP, 60,118,62,8
LTEXT           "Character Set:", IDC_STATIC, 8,137,46,8
EDITTEXT        IDC_LF_CHARSET, 58,135,24,12, ES_AUTOHSCROLL
PUSHBUTTON      "?", IDC_CHARSET_HELP, 90,135,14,14
GROUPBOX        "Quality", IDC_STATIC, 132,98,62,48, WS_GROUP
CONTROL         "Default", IDC_DEFAULT_QUALITY, "Button",
                BS_AUTORADIOBUTTON, 136,110,40,8
CONTROL         "Draft", IDC_DRAFT_QUALITY, "Button", BS_AUTORADIOBUTTON,
                136,122,40,8
CONTROL         "Proof", IDC_PROOF_QUALITY, "Button", BS_AUTORADIOBUTTON,
                136,134,40,8
LTEXT           "Face Name:", IDC_STATIC, 8,154,44,8
EDITTEXT        IDC_LF_FACENAME, 58,152,136,12, ES_AUTOHSCROLL
GROUPBOX        "Output Precision", IDC_STATIC, 8,166,118,133, WS_GROUP
CONTROL         "OUT_DEFAULT_PRECIS", IDC_OUT_DEFAULT, "Button",
                BS_AUTORADIOBUTTON, 12,178,112,8
CONTROL         "OUT_STRING_PRECIS", IDC_OUT_STRING, "Button",
                BS_AUTORADIOBUTTON, 12,191,112,8
CONTROL         "OUT_CHARACTER_PRECIS", IDC_OUT_CHARACTER, "Button",
                BS_AUTORADIOBUTTON, 12,204,112,8
CONTROL         "OUT_STROKE_PRECIS", IDC_OUT_STROKE, "Button",
                BS_AUTORADIOBUTTON, 12,217,112,8
```

```
CONTROL         "OUT_TT_PRECIS",IDC_OUT_TT,"Button",BS_AUTORADIOBUTTON,
                12,230,112,8
CONTROL         "OUT_DEVICE_PRECIS",IDC_OUT_DEVICE,"Button",
                BS_AUTORADIOBUTTON,12,243,112,8
CONTROL         "OUT_RASTER_PRECIS",IDC_OUT_RASTER,"Button",
                BS_AUTORADIOBUTTON,12,256,112,8
CONTROL         "OUT_TT_ONLY_PRECIS",IDC_OUT_TT_ONLY,"Button",
                BS_AUTORADIOBUTTON,12,269,112,8
CONTROL         "OUT_OUTLINE_PRECIS",IDC_OUT_OUTLINE,"Button",
                BS_AUTORADIOBUTTON,12,282,112,8
GROUPBOX        "Pitch",IDC_STATIC,132,166,62,50,WS_GROUP
CONTROL         "Default",IDC_DEFAULT_PITCH,"Button",BS_AUTORADIOBUTTON,
                137,176,52,8
CONTROL         "Fixed",IDC_FIXED_PITCH,"Button",BS_AUTORADIOBUTTON,137,
                189,52,8
CONTROL         "Variable",IDC_VARIABLE_PITCH,"Button",
                BS_AUTORADIOBUTTON,137,203,52,8
GROUPBOX        "Family",IDC_STATIC,132,218,62,82,WS_GROUP
CONTROL         "Don't Care",IDC_FF_DONTCARE,"Button",BS_AUTORADIOBUTTON,
                137,229,52,8
CONTROL         "Roman",IDC_FF_ROMAN,"Button",BS_AUTORADIOBUTTON,137,241,
                52,8
CONTROL         "Swiss",IDC_FF_SWISS,"Button",BS_AUTORADIOBUTTON,137,253,
                52,8
CONTROL         "Modern",IDC_FF_MODERN,"Button",BS_AUTORADIOBUTTON,137,
                265,52,8
CONTROL         "Script",IDC_FF_SCRIPT,"Button",BS_AUTORADIOBUTTON,137,
                277,52,8
CONTROL         "Decorative",IDC_FF_DECORATIVE,"Button",
                BS_AUTORADIOBUTTON,137,289,52,8
DEFPUSHBUTTON   "OK",IDOK,247,286,50,14
GROUPBOX        "Text Metrics",IDC_STATIC,201,2,140,272,WS_GROUP
LTEXT           "Height:",IDC_STATIC,207,12,64,8
LTEXT           "0",IDC_TM_HEIGHT,281,12,44,8
LTEXT           "Ascent:",IDC_STATIC,207,22,64,8
LTEXT           "0",IDC_TM_ASCENT,281,22,44,8
LTEXT           "Descent:",IDC_STATIC,207,32,64,8
LTEXT           "0",IDC_TM_DESCENT,281,32,44,8
LTEXT           "Internal Leading:",IDC_STATIC,207,42,64,8
LTEXT           "0",IDC_TM_INTLEAD,281,42,44,8
LTEXT           "External Leading:",IDC_STATIC,207,52,64,8
LTEXT           "0",IDC_TM_EXTLEAD,281,52,44,8
LTEXT           "Ave Char Width:",IDC_STATIC,207,62,64,8
LTEXT           "0",IDC_TM_AVECHAR,281,62,44,8
```

```
    LTEXT           "Max Char Width:",IDC_STATIC,207,72,64,8
    LTEXT           "0",IDC_TM_MAXCHAR,281,72,44,8
    LTEXT           "Weight:",IDC_STATIC,207,82,64,8
    LTEXT           "0",IDC_TM_WEIGHT,281,82,44,8
    LTEXT           "Overhang:",IDC_STATIC,207,92,64,8
    LTEXT           "0",IDC_TM_OVERHANG,281,92,44,8
    LTEXT           "Digitized Aspect X:",IDC_STATIC,207,102,64,8
    LTEXT           "0",IDC_TM_DIGASPX,281,102,44,8
    LTEXT           "Digitized Aspect Y:",IDC_STATIC,207,112,64,8
    LTEXT           "0",IDC_TM_DIGASPY,281,112,44,8
    LTEXT           "First Char:",IDC_STATIC,207,122,64,8
    LTEXT           "0",IDC_TM_FIRSTCHAR,281,122,44,8
    LTEXT           "Last Char:",IDC_STATIC,207,132,64,8
    LTEXT           "0",IDC_TM_LASTCHAR,281,132,44,8
    LTEXT           "Default Char:",IDC_STATIC,207,142,64,8
    LTEXT           "0",IDC_TM_DEFCHAR,281,142,44,8
    LTEXT           "Break Char:",IDC_STATIC,207,152,64,8
    LTEXT           "0",IDC_TM_BREAKCHAR,281,152,44,8
    LTEXT           "Italic?",IDC_STATIC,207,162,64,8
    LTEXT           "0",IDC_TM_ITALIC,281,162,44,8
    LTEXT           "Underlined?",IDC_STATIC,207,172,64,8
    LTEXT           "0",IDC_TM_UNDER,281,172,44,8
    LTEXT           "Struck Out?",IDC_STATIC,207,182,64,8
    LTEXT           "0",IDC_TM_STRUCK,281,182,44,8
    LTEXT           "Variable Pitch?",IDC_STATIC,207,192,64,8
    LTEXT           "0",IDC_TM_VARIABLE,281,192,44,8
    LTEXT           "Vector Font?",IDC_STATIC,207,202,64,8
    LTEXT           "0",IDC_TM_VECTOR,281,202,44,8
    LTEXT           "TrueType Font?",IDC_STATIC,207,212,64,8
    LTEXT           "0",IDC_TM_TRUETYPE,281,212,44,8
    LTEXT           "Device Font?",IDC_STATIC,207,222,64,8
    LTEXT           "0",IDC_TM_DEVICE,281,222,44,8
    LTEXT           "Family:",IDC_STATIC,207,232,64,8
    LTEXT           "0",IDC_TM_FAMILY,281,232,44,8
    LTEXT           "Character Set:",IDC_STATIC,207,242,64,8
    LTEXT           "0",IDC_TM_CHARSET,281,242,44,8
    LTEXT           "0",IDC_TM_FACENAME,207,262,128,8
END

////////////////////////////////////////////////////////////////////////////
// Menu

PICKFONT MENU DISCARDABLE
BEGIN
```

```
    POPUP "&Device"
    BEGIN
        MENUITEM "&Screen",          IDM_DEVICE_SCREEN, CHECKED
        MENUITEM "&Printer",         IDM_DEVICE_PRINTER
    END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by PickFont.rc

#define IDC_LF_HEIGHT                   1000
#define IDC_LF_WIDTH                    1001
#define IDC_LF_ESCAPE                   1002
#define IDC_LF_ORIENT                   1003
#define IDC_LF_WEIGHT                   1004
#define IDC_MM_TEXT                     1005
#define IDC_MM_LOMETRIC                 1006
#define IDC_MM_HIMETRIC                 1007
#define IDC_MM_LOENGLISH                1008
#define IDC_MM_HIENGLISH                1009
#define IDC_MM_TWIPS                    1010
#define IDC_MM_LOGTWIPS                 1011
#define IDC_LF_ITALIC                   1012
#define IDC_LF_UNDER                    1013
#define IDC_LF_STRIKE                   1014
#define IDC_MATCH_ASPECT                1015
#define IDC_ADV_GRAPHICS                1016
#define IDC_LF_CHARSET                  1017
#define IDC_CHARSET_HELP                1018
#define IDC_DEFAULT_QUALITY             1019
#define IDC_DRAFT_QUALITY               1020
#define IDC_PROOF_QUALITY               1021
#define IDC_LF_FACENAME                 1022
#define IDC_OUT_DEFAULT                 1023
#define IDC_OUT_STRING                  1024
#define IDC_OUT_CHARACTER               1025
#define IDC_OUT_STROKE                  1026
#define IDC_OUT_TT                      1027
#define IDC_OUT_DEVICE                  1028
#define IDC_OUT_RASTER                  1029
#define IDC_OUT_TT_ONLY                 1030
#define IDC_OUT_OUTLINE                 1031
#define IDC_DEFAULT_PITCH               1032
```

```
# define IDC_FIXED_PITCH          1033
# define IDC_VARIABLE_PITCH       1034
# define IDC_FF_DONTCARE          1035
# define IDC_FF_ROMAN             1036
# define IDC_FF_SWISS             1037
# define IDC_FF_MODERN            1038
# define IDC_FF_SCRIPT            1039
# define IDC_FF_DECORATIVE        1040
# define IDC_TM_HEIGHT            1041
# define IDC_TM_ASCENT            1042
# define IDC_TM_DESCENT           1043
# define IDC_TM_INTLEAD           1044
# define IDC_TM_EXTLEAD           1045
# define IDC_TM_AVECHAR           1046
# define IDC_TM_MAXCHAR           1047
# define IDC_TM_WEIGHT            1048
# define IDC_TM_OVERHANG          1049
# define IDC_TM_DIGASPX           1050
# define IDC_TM_DIGASPY           1051
# define IDC_TM_FIRSTCHAR         1052
# define IDC_TM_LASTCHAR          1053
# define IDC_TM_DEFCHAR           1054
# define IDC_TM_BREAKCHAR         1055
# define IDC_TM_ITALIC            1056
# define IDC_TM_UNDER             1057
# define IDC_TM_STRUCK            1058
# define IDC_TM_VARIABLE          1059
# define IDC_TM_VECTOR            1060
# define IDC_TM_TRUETYPE          1061
# define IDC_TM_DEVICE            1062
# define IDC_TM_FAMILY            1063
# define IDC_TM_CHARSET           1064
# define IDC_TM_FACENAME          1065
# define IDM_DEVICE_SCREEN        40001
# define IDM_DEVICE_PRINTER       40002
```

图 17-1 显示了典型的 PICKFONT 屏幕显示。PICKFONT 左半部分显示了一个无模式对话框，通过它可以选择逻辑字体结构的大部分字段。对话框的右半部分显示了字体选入设备描述表后 GetTextMetrics 的结果。对话框的下部，程序使用这种字体显示一个字符串。因为非模态对话框非常大，所以最好不要在 1024 × 768 或更大的显示尺寸下运行这个程序。

非模态对话框还包含一些非逻辑字体结构的选项，它们是包括 Logical Twips 方式

的映射方式、Match Aspect 选项(更改 Windows 将逻辑字体与真实字体匹配的方式)和 Adv Grtx Mode(设置 Windows NT 中的高级图形模式)。稍后我将对这些做详细讨论。

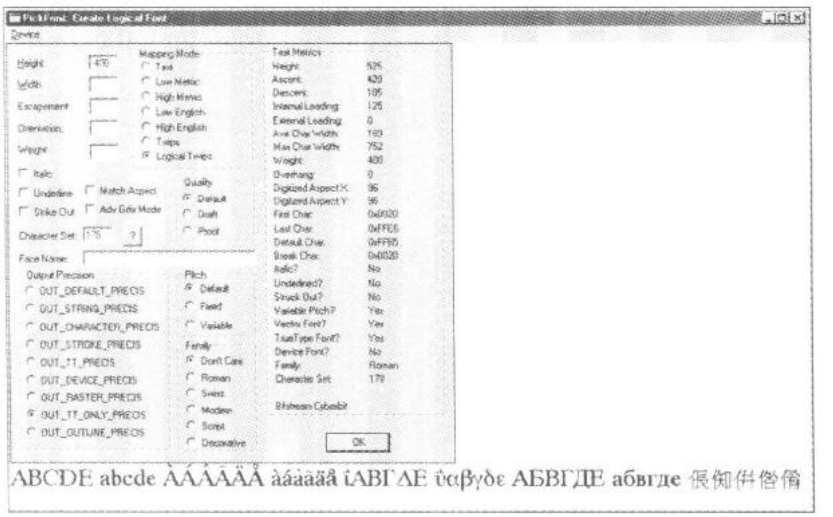


图 17-1　典型的 PICKFONT 显示(Windows NT 下的 Unicode 版本)

从 Device 菜单中,可以选择默认打印机而不是视频显示器。在这种情况下,PICKFONT 将逻辑字体选入打印机设备描述表中,并从打印机显示 TEXTMETRIC 结构。然后,程序将逻辑字体选入窗口设备描述表中,以显示样本字符串。因此,程序显示的文本可能会使用与 TEXTMETRIC 字段所描述的字体(打印机字体)不同的字体(屏幕字体)。

PICKFONT 程序包含了保存对话框的必要逻辑,因此我不会详细讨论该程序的工作方式。当您创建和选择逻辑字体时,我将解释其中的原理。

17.3.3　逻辑字体结构

可以调用 CreateFont 来创建逻辑字体,它是具有 14 个参数的函数。一般地,先定义一个 LOGFONT 类型的结构

```
LOGFONT lf ;
```

然后定义该结构的字段会更容易一些。完成后,可以使用指向该结构的指针调用 CreateFontIndirect:

```
hFont = CreatFontIndirect (&lf) ;
```

不必设置 LOGFONT 结构的每个字段。如果逻辑字体结构定义为静态变量,那么所

有的字段都会初始化为0,0一般是默认值。然后,可以不用更改而直接使用这个结构,CreateFontIndirect会返回字体的句柄。当您将该字体选入设备描述表时,会得到一个合理的默认字体。您可以根据自己的需要,明确或模糊地填充LOGFONT结构,Windows会用一种真实字体与您的要求相匹配。

在讨论LOGFONT结构中的每个字段时,您可能想用PICKFONT程序来测试它们。当您希望程序使用输入的任何字段时,别忘了按Enter键或OK按钮。

LOGFONT结构的前两个字段是逻辑单位,因此它们依赖于映射方式的当前设置:

- lfHeight 这是以逻辑单位表示的所期望的字符高度。您可以将lfHeight设置0,以使用默认大小,或者根据字段代表的含义将其设置为正数或负数。如果将lfHeight设置为正数,就表示您希望该值表示含有内部间隔(不是外部间隔)的高度。实际上,要求的字体行距为lfHeight。如果将lfHeight设置为负值,则Windows会将其绝对值作为与磅值一致的字体高度。这是一个很重要的区别:如果想要特定磅值的字体,可将磅值转换为逻辑单位,并将lfHeight字段设置为该值的负数。如果lfHeight是正值,则TEXTMETRIC结构的tmHeight字段近似为该值(有时有微小的偏差,可能是由于舍入误差所引起)。如果lfHeight是负值,则它粗略地与不包括tmInternalLeading字段的TEXTMETRIC结构的tmHeight字段相匹配。
- lfWidth 这是以逻辑单位表示的所期望的字符宽度。在多数情况下,可以将此值设置为0,让Windows仅根据高度选择字体。使用非零值对点阵字体并不起太大作用,但对于TrueType字体,您能轻松地用它来获得比正常字符更宽或更窄的字体。这个字段对应于TEXTMETRIC结构的tmAveCharWidth字段。要正确使用lfWidth字段,首先把带有lfWidth字段的LOGFONT结构设置为0,创建逻辑字体,将它选入设备描述表,然后调用GetTextMetrics。得到tmAveCharWidth字段,可按比例调节其值的大小,然后使用所调节的lfWidth的tmAveCharWidth值来创建第二种字体。

下两个字段:lfEscapement和lfOrientation指定文本的"移位"和"方向"。理论上,lfEscapement使字符串能够以一定的角度书写(但每个字符的基线仍与水平轴平行),而lfOrientation使单个字符倾斜。但是这两个字段并不是那么有效,甚至今天它们也不过只有在下面的情况下才能很好地起作用:使用TureType字体、运行Windows NT,以及首先用CM_ADVANCED标志设置调用SetGraphicsMode。通过选中Adv Grfx Mode复选框,您可以完成PICKFONT中的最终需要。

在PICKFONT中验证这些字段时,要注意单位是1/10度,逆时针方向旋转。它很容易输入一个值使示例文本串消失!因此,请使用-600~0或3000~3600之间的值。

- lfEscapement 这是从水平方向上逆时针测量的十分之几的角度。它指定在书写文本时字符串的连续字符放置的方式。表17-1给出了几个例子。在Windows 98中,这个值设置了TrueType文本的移位和方向。在Windows NT中,这个值通常也是这

样设置,除了用 GM_ADVANCED 参数调用 SetGraphicsMode 时,它按文档中说明的那样工作。

- lfOrientation　这是从水平方向逆时针测量的十分之几的角度,它影响单个字符的外观。表 17-2 给出了几个例子。这个字段一般不起作用,除非在 Windows NT 下使用 TrueType 字体并把图像模式设置为 GM_ADVANCED 时,它才按文档中说明的那样工作。

表 17-1

值	字符的放置
0	从左向右(默认)
900	向上
1800	从右向左
2700	向下

表 17-2

值	字符外观
0	正常(默认)
900	向右倾斜 90°
1800	颠倒
2700	向左倾斜 90°

其余 10 个字段如下:

- lfWeight　这个字段使您能够指定粗体。WINGDI.H 头文件定义了可用于这个字段的一组值(参见表 17-3)。事实上,它比以前用过的任何一组值都完善。您可以对标准字使用 0 或 400,对粗体使用 700。

表 17-3

值	标识符
0	FW_DONTCARE
100	FW_THIN
200	FW_EXTRALIGHT 或 FW_ULTRALIGHT
300	FW_LIGHT
400	FW_NORMAL 或 FW_REGULAR
500	FW_MEDIUM
600	FW_SEMIBOLD 或 FW_DEMIBOLD
700	FW_BOLD
800	FW_EXTRABOLD 或 FW_ULTRABOLD
900	FW_HEAVY 或 FW_BLACK

- lfItalic 在非零值时,它指定斜体。Windows 能在 GDI 点阵字体上合成斜体。即 Windows 仅仅移动若干行字符位图来模仿斜体。对于 TrueType 字体,Windows 使用真正的斜体或字体的倾斜版本。
- lfUnderline 在非零值时,它指定下划线,始终是在 GDI 字体上合成的。也就是说,Windows GDI 简单地在包括空格在内的每个字符下划线。
- lfStrikeOut 在非零值时,它指定字体上应该有一条线穿过。这也是由 GDI 字体合成的。
- lfCharSet 这是指定字体字符集的一个字节的值。我会在下一节"字符集和 Unicode"中更详细地讨论这个字段。在 PICKFONT 中,您可以按下带有问号的按钮来获取能够使用的字符集列表。

  注意,lfCharSet 字段是唯一不用 0 表示默认值的字段。0 相当于 ANSI_CHARSET,ANSI 字符在美国和西欧使用。DEFAULT_CHARSET 代码等于 1,表示程序运行的机器上默认的字符集。
- lfOutPrecision 它指定了 Windows 用实际的字体匹配期望的字体大小和特征的方式。这是一个复杂的字段,一般很少使用。请查看关于 LOGFONT 结构的文档,以得到更详细的信息。注意,可以使用 OUT_TT_ONLY_PRECIS 标志来确保得到的是 TrueType 字体。
- lfClipPrecision 这个字段指定了当字符的一部分位于剪裁区以外时,剪裁字符的方式。这个字段不经常使用,PICKFONT 程序也没有使用它。
- lfQuality 这是一条关于期望字体与实际字体相匹配的 Windows 指令。它实际只对点阵字体有意义,并不影响 TrueType 字体。DRAFT_QUALITY 标志指出需要 GDI 缩放点阵字体以得到想要的大小;PROOF_QUALITY 标志指出不需缩放。PROOF_QUALITY 字体最漂亮,但它们可能比所希望的要小一些。这个字段中也可以使用 DEFAULT_QUALITY(或 0)。
- lfPitchAndFamily 这个字节由两部分组成。您可以使用按位或运算符结合用于此字段的两个标识符。最低的两位指定字体是等宽字体(所有字符的宽度相等)还是变宽字体(参见表 17-4)。

表 17-4

值	标识符
0	DEFAULT_PITCH
1	FIXED_PITCH
2	VARIABLE_PITCH

字节的上半部分指定字体族(参见表 17-5)。

表 17-5

值	标识符
0x00	FW_DONTCARE
0x10	FF_ROMAN(变宽,serifs)
0x20	FF_SWISS(变宽,非 serifs)
0x30	FF_MODERN(定间距)
0x40	FF_SCRIPT(模仿手写)
0x50	FF_DECORATIVE

- lfFaceName　这是关于字样(如 Courier、Arial 或 Times New Roman)的实际的文本名。这个字段是宽度为 LF_FACESIZE(或 32 个字符)的字节数组。如果要得到 TrueType 的斜体或粗体字体,有两种方法。在 lfFaceName 字段中使用完整的字样名(如 Times New Roman Italic),或者可以使用基本名称(即 Times New Roman),并设置 lfItalic 字段。

17.3.4　字体映射算法

在设置了逻辑字体结构后,调用 CreateFontIndirect 来得到逻辑字体句柄。当调用 SelectObject 把逻辑字体选入设备描述表时,Windows 查找与所需字体最接近匹配的实际字体。它使用"字体映射算法"。结构的某些字段要比其他字段更重要一些。

了解字体映射的最好方式是花一些时间试验 PICKFONT。以下是几条指南:

- lfCharSet(字符集)字段是非常重要的。如果指定了 OEM_CHARSET(255),会得到某种笔划字体或终端字体,因为它们是仅有的使用 OEM 字符集的字体。然而,随着 TrueType"Big Fonts"的出现(在本书 6.4.5 节中讨论过),单一的 TrueType 字体可以映射到包括 OEM 字符集等不同的字符集。您需要使用 SYMBOL_CHARSET(2) 来得到 Symbol 字体或 Wingdings 字体。
- lfPitchAndFamily 字段的 FIXED_PITCH 间距值很重要,因为实际上是告诉 Windows 不想处理变宽字体。
- lfFaceName 字段很重要,因为指定了所需字体的字样。如果将 lfFaceName 设置为 NULL,并在 lfPitchAndFamily 字段中将组值设置为 FF_DONTCARE 以外的值,因为指定了字体族,所以该字段也很重要。
- 对于点阵字体,Windows 会试图匹配 lfHeight 值,即使需要增加较小字体的大小。实际字体的高度总是小于或等于所需的字体,除非没有更小的字体满足您的要求。对于笔划字体或 TrueType 字体,Windows 只是简单地将字体缩放到需要的高度。

- 可以通过将 lfQuality 设置为 PROOF_QUALITY 来防止 Windows 缩放点阵字体。这么做,就告诉 Windows 所需的字体高度没有字体外观重要。
- 如果指明了对于显示器的特定纵横比不协调的 lfHeight 和 lfWeight 值,Windows 可以映射到为显示器或其他不同纵横比的设备设计的点阵字体。这是得到细或粗字体的技巧。(当然,对于 TrueType 字体是不必要的。)一般情况下,可能想避免为另一种设备匹配字体,可以通过单击 Match Aspect 复选框,在 PICKFONT 中完成。如果选中了该复选框,那么 PICKFONT 使用 TRUE 参数调用 SetMapperFlags。

17.3.5 查找关于字体的信息

在 PICKFONT 中非模态对话框的右侧是字体选入设备描述表后从 GetTextMetrics 函数中获得的信息。(注意,可以使用 PICKFONT 的 Device 菜单指出设备描述表是屏幕还是默认打印机。因为在打印机上有效的字体可能不同,所以结果也可能不同。)在 PICKFONT 列表的底部是从 GetTextFace 得到的有效字样名。

除了数字化的纵横比以外,Windows 复制到 TEXTMETRIC 结构的所有大小值都是逻辑单位。TEXTMETRIC 结构的字段如下:

- tmHeight 逻辑单位的字符高度。它近似等于 LOGFONT 结构中指定的 lfHeight 字段的值,如果该值为正,它就代表行距,而非磅值。如果 LOGFONT 结构的 lfHeight 字段为负,则 tmHeight 字段减去 tmInternalLeading 字段应近似等于 lfHeight 字段的绝对值。
- tmAscent 逻辑单位的基线以上的字符垂直大小。
- tmDescent 逻辑单位的基线以下的字符垂直大小。
- tmInternalLeading 包含在 tmHeight 值内的垂直大小,通常被一些大写字母上注音符号占据。同样,可以用 tmHeight 值减去 tmInternalLeading 值来计算字体的磅值。
- tmExternalLeading tmHeight 以外的行距附加量,字体的设计者推荐用于隔开文本的连续行。
- tmAveCharWidth 字体中小写字母的平均宽度。
- tmMaxCharWidth 逻辑单位的字符最大宽度。对于等宽字体,这个值与 tmAveCharWidth 相同。
- tmWeight 字体粗细,范围为 0 ~ 999。实际上,这个字段为 400 时是标准字体,700 时是粗体。
- tmOverhang Windows 在合成斜体或粗体时添加到点阵字体字符的额外宽度(逻辑单位)。当点阵字体斜体化时,tmAveCharWidth 值保持不变,因为斜体化的字符串与相同的正常字符串的总宽度相等。要给字体加粗,Windows 必须稍微增加每个字符的宽度。对于粗体,tmAveCharWidth 值小于 tmOverhang 值,等于没有加粗的相同字体的 tmAveCharWidth 值。

- tmDigitizedAspectX 和 tmDigitizedAspectY　字体合适的纵横比。它们与使用 LOGPIXELSX 和 LOGPIXELSY 标识符从 GetDeviceCaps 得到的值相同。
- tmFirstChar　字体中第一个字符的字符代码。
- tmLastChar　字体中最后一个字符的字符代码。如果 TEXTMETRIC 结构通过调用 GetTextMetricsW(函数的宽字符版本)获得,那么这个值可能大于 255。
- tmDefaultChar　Windows 用于显示字体中没有的字符的字符代码,通常是矩形。
- tmBreakChar　在调整文本时,Windows 和您的程序用于分隔单词的字符。如果您不用一些奇怪的东西(例如 EBCDIC 字体),它就是 32——空格字符。
- tmItalic　对于斜体字为非零值。
- tmUnderlined　对于下划线字体为非零值。
- tmStruckOut　对于删除线字体为非零值。
- tmPitchAndFamily　低 4 位是表示字体某些特征的标志,由在 WINGDI.H 中定义的标识符指出(参见表 17-6)。

表 17-6

值	标识符
0x01	TMPF_FIXED_PITCH
0x02	TMPF_VECTOR
0x04	TMPF_TRUETYPE
0x08	TMPF_DEVICE

不管上表中标识符 TMPF_FIXED_PITCH 标志的名称是什么,如果字体字符是变宽的,则最低位为 1。第二最低位(TMPF_VECTOR)对于 TrueType 字体和使用其他可缩放的轮廓技术的字体(如 PostScript 的字体)也为 1。TMPF_DEVICE 标志表示设备字体(即打印机内置的字体),而不是基于 GDI 的字体。这个字段的高 4 位表示字体族,并且与在 LOGFONT 的 lfPitchAndFamily 字段中所用的值相同。

- tmCharSet　字符集标识符。

17.3.6　字符集和 Unicode

我在第六章讨论了 Windows 字符集的概念,那里我们必须处理涉及键盘的国际化问题。在 LOGFONT 和 TEXTMETRIC 结构中,所需字体(或实际字体)的字符集由 0 ~ 255 之间的单个字节的数值表示。定义在 WINGDI.H 中的字符集标识符如下所示:

```
# define ANSI_CHARSET            0
# define DEFAULT_CHARSET         1
```

```
# define SYMBOL_CHARSET          2
# define MAC_CHARSET            77
# define SHIFTJIS_CHARSET      128
# define HANGEUL_CHARSET       129
# define HANGUL_CHARSET        129
# define JOHAB_CHARSET         130
# define GB2312_CHARSET        134
# define CHINESEBIG5_CHARSET   136
# define GREEK_CHARSET         161
# define TURKISH_CHARSET       162
# define VIETNAMESE_CHARSET    163
# define HEBREW_CHARSET        177
# define ARABIC_CHARSET        178
# define BALTIC_CHARSET        186
# define RUSSIAN_CHARSET       204
# define THAI_CHARSET          222
# define EASTEUROPE_CHARSET    238
# define OEM_CHARSET           255
```

字符集与代码页的概念类似,但是字符集特定于 Windows,通常小于或等于 255。

与本书的所有程序一样,您可以使用定义的 UNICODE 标识符来编译 PICKFONT,也可以不使用 UNICODE 标识符来编译它。和往常一样,本书配套盘上程序的两个版本分别位于 DEBUG 和 RELEASE 目录中。

注意,在程序的 Unicode 版本中,PICKFONT 在其窗口底部显示的字符串要更长一些。在两个版本中,字符串的字符代码由 0x40 ~ 0x45、0x60 ~ 0x65。无论选择了哪种字符集(除了 SYMBOL_CHARSET),这些字符代码都显示拉丁字母表的前 5 个大写和小写字母(即 A ~ E 和 a ~ e)。

当运行 PICKFONT 程序的非 Unicode 版本时,接下来的 12 个字符——字符代码 0xC0 ~ 0xC5 和 0xE0 ~ 0xE5——将依赖于选定的字符集。对于 ANSI_CHARSET,这个字符代码对应于字母 A 大小写的加重音版本。对于 GREEK_CHARSET,这些代码对应于希腊字母表的字母。对于 RUSSIAN_CHARSET,对应于西里尔字母表的字母。注意,当您选择一种字符集时,字体可能会改变,这是因为点阵字体可能没有这些字符,但 TrueType 字体可能有。您可能回忆起大多数 TrueType 字体是"Big fonts",并且包含几种不同字符集的字母。如果运行 Windows 的远东版本,那么这些字符会被解释为双字节字符,并且会按表意符号显示,而不是按字母显示。

在 Windows NT 下运行 PICKFONT 的 Unicode 版本时,代码 0xC0 ~ 0xC5 和 0xE0 ~ 0xE5 通常是字母 A 大小写的加重音版本(除了 SYMBOL_CHARSET),因为 Unicode 中定义了这些代码。程序也显示 0x0390 ~ 0x0395 和 0x03B0 ~ 0x03B5 的字符代码。由于它

们在 Unicode 中有定义，所以这些代码总是对应于希腊字母表的字母。类似地，程序显示 0x0410 ~ 0x0415 和 0x0430 ~ 0x0435 的字符代码，它们对应于西里尔字母表的字母。然而，请注意这些字符也许不会存在于默认字体中，您必须选择 GREEK_CHARSET 或 RUSSIAN_CHARSET 来得到它们。在这种情况下，LOGFONT 结构中的字符集 ID 不会更改实际的字符集，字符集总是 Unicode。而字符集 ID 指出来自该字符集的字符是所需的。

现在选择 HEBREW_CHARSET(代码 177)。希伯来字母表不包括在 Windows 通常的 Big Fonts 中，因此操作系统选择 Lucida Sans Unicode，这一点您可以在非模态对话框的右下角中验证。

PICKFONT 也显示 0x5000 ~ 0x5004 的字符代码，它们对应于汉语、日语和朝鲜语象形文字的大多数部分。如果运行 Windows 的远东版本，或者下载了比 Lucida Sans Unicode 范围更广的免费 Unicode 字体，那么您就可以看到这些字符。Bitstream CyberBit 字体就是这样的一种字体，可以从 http://www.bitstream.com/products/world/cyberbits 中找到。(Lucida Sans Unicode 大约有 300 KB，而 Bitstream CyberBit 有大约 13 MB。)如果安装了这种字体，当需要一种 Lucida Sans Unicode 不支持的字体时，Windows 会选择它，这些字体如：SHIFTJIS_CHARSET (日语)、HANGUL_CHARSET(朝鲜语)、JOHAB_CHARSET (朝鲜语)、GB2312_CHARSET(简体中文)，或 CHINESEBIG5_CHARSET (繁体中文)。

本章的后面有一个程序，允许您查看 Unicode 字体的所有字母。

17.3.7 EZFONT 系统

TrueType 字体系统(以传统的排版为基础)为 Windows 以不同的方式显示文本提供了牢固的基础。但是一些 Windows 的字体选择函数基于老式的技术，点阵字体接近于打印机设备字体。下一节将讲到字体列举，它能够使程序获得显示器或打印机上全部有效字体的列表。不过，“ChooseFont”对话框(一会儿讨论)在很大程度上消除了程序列举字体的必要性。

因为标准的 TrueType 字体可以在任何系统上使用，并且这些字体既可以用于显示器，也可以用于打印机，所以程序为了选择 TrueType 字体而列举字体，或者盲目地要求某种与所需字体相似的字体，都是没有必要的。程序可以简单明确地选择系统中存在的 TrueType 字体(当然，除非用户有意地删除了它们)。事实上，这种方法几乎与指定字体名称(可能为 17.2.2“TrueType 字体”节中列出的 13 种字体中的一种)和字体大小一样简单。我把这种方法称作 EZFONT(“简便字体”)，下页所示的程序 17-2 列出了它的两个文件。

程序 17-2　EZFONT 文件

EZFONT.H

```
/*------------
    EZFONT.H header file
  ------------*/

HFONT EzCreateFont (HDC hdc, TCHAR * szFaceName, int iDeciPtHeight,
                    int iDeciPtWidth, int iAttributes, BOOL fLogRes) ;
#define EZ_ATTR_BOLD            1
#define EZ_ATTR_ITALIC          2
#define EZ_ATTR_UNDERLINE       4
#define EZ_ATTR_STRIKEOUT       8
```

EZFONT.C

```
/*------------------------------------------
    EZFONT.C——Easy Font Creation
              (c) Charles Petzold, 1998
  ------------------------------------------*/

#include <windows.h>
#include <math.h>
#include "ezfont.h"

HFONT EzCreateFont (HDC hdc, TCHAR * szFaceName, int iDeciPtHeight,
                    int iDeciPtWidth, int iAttributes, BOOL fLogRes)
{
     FLOAT      cxDpi, cyDpi ;
     HFONT      hFont ;
     LOGFONT    lf ;
     POINT      pt ;
     TEXTMETRIC tm ;

     SaveDC (hdc) ;

     SetGraphicsMode (hdc, GM_ADVANCED) ;
     ModifyWorldTransform (hdc, NULL, MWT_IDENTITY) ;
     SetViewportOrgEx (hdc, 0, 0, NULL) ;
     SetWindowOrgEx   (hdc, 0, 0, NULL) ;

     if (fLogRes)
     {
          cxDpi = (FLOAT) GetDeviceCaps (hdc, LOGPIXELSX) ;
          cyDpi = (FLOAT) GetDeviceCaps (hdc, LOGPIXELSY) ;
     }
     else
     {
```

```
        cxDpi = (FLOAT) (25.4 * GetDeviceCaps (hdc, HORZRES) /
                                GetDeviceCaps (hdc, HORZSIZE)) ;

        cyDpi = (FLOAT) (25.4 * GetDeviceCaps (hdc, VERTRES) /
                                GetDeviceCaps (hdc, VERTSIZE)) ;
    }

    pt.x = (int) (iDeciPtWidth * cxDpi / 72) ;
    pt.y = (int) (iDeciPtHeight * cyDpi / 72) ;

    DPtoLP (hdc, &pt, 1) ;
    lf.lfHeight         = - (int) (fabs (pt.y) / 10.0 + 0.5) ;
    lf.lfWidth          = 0 ;
    lf.lfEscapement     = 0 ;
    lf.lfOrientation    = 0 ;
    lf.lfWeight         = iAttributes & EZ_ATTR_BOLD        ?  700 : 0 ;
    lf.lfItalic         = iAttributes & EZ_ATTR_ITALIC      ?    1 : 0 ;
    lf.lfUnderline      = iAttributes & EZ_ATTR_UNDERLINE   ?    1 : 0 ;
    lf.lfStrikeOut      = iAttributes & EZ_ATTR_STRIKEOUT   ?    1 : 0 ;
    lf.lfCharSet        = DEFAULT_CHARSET ;
    lf.lfOutPrecision   = 0 ;
    lf.lfClipPrecision  = 0 ;
    lf.lfQuality        = 0 ;
    lf.lfPitchAndFamily = 0 ;

    lstrcpy (lf.lfFaceName, szFaceName) ;

    hFont = CreateFontIndirect (&lf) ;

    if (iDeciPtWidth != 0)
    {
        hFont = (HFONT) SelectObject (hdc, hFont) ;

        GetTextMetrics (hdc, &tm) ;

        DeleteObject (SelectObject (hdc, hFont)) ;

        lf.lfWidth = (int) (tm.tmAveCharWidth *
                            fabs (pt.x) / fabs (pt.y) + 0.5) ;

        hFont = CreateFontIndirect (&lf) ;
    }

    RestoreDC (hdc, -1) ;
    return hFont ;
}
```

EZFONT.C 只有一个函数，称为 EzCreateFont，如下所示：

```
hFont = EzCreateFont (hdc, szFaceName, iDeciPtHeight, iDeciPtWidth,
                      iAttributes, fLogRes) ;
```

该函数返回一个字体句柄。可以通过调用 SelectObject 将该字体选入设备描述表，然后调用 GetTextMetrics 或 GetOutlineTextMetrics 以确定字体尺寸在逻辑坐标中的实际大小。在程序终止前，应该调用 DeleteObject 来删除任何创建的字体。

szFaceName 参数可以是任意的 TrueType 字体名。选定的字体越接近标准字体，该字体在系统中存在的概率就越大。

第 3 个参数指出了所需的磅值，但是它的单位是 1/10 磅。因而，如果所需要的磅值为12.5，则该参数应取值为 125。

第 4 个参数通常应设置为 0，或与第 3 个参数相同。然而，通过将此字段设置为不同值可以创建更宽或更窄的 TrueType 字体。它以磅为单位描述了字体的宽度，有时称之为字体的“全宽(em-width)”。不要将它与字体字符的平均宽度或其他类似的东西相混淆。在过去的排版技术中，大写字母 M 的宽度与高度是相等的。于是，“完全正方形(em-square)”的概念产生了，这是全宽测量的起源。当字体的全宽等于字体的全高(字体的磅值)时，字符宽度是字体设计者设定的宽度。宽或窄的全宽值可以产生更细或更宽的字符。

可以将 iAttributes 参数设置为以下定义在 EZFONT.H 中的值：

```
EZ_ATTR_BOLD
EZ_ATTR_ITALIC
EZ_ATTR_UNDERLINE
EZ_ATTR_STRIKEOUT
```

可以使用 EZ_ATTR_BOLD 或 EZ_ATTR_ITALIC，或者将风格作为完整 TrueType 字体名的一部分。

最后，我们将参数 fLogRes 设置为逻辑值 TRUE，以表示字体磅值与设备的“逻辑分辨率”相吻合，其中“逻辑分辨率”是 GetDeviceCaps 函数使用 LOGPIXELSX 和 LOGPIXELSY 参数的返回值。另外，基于分辨率的字体大小是从 HORZRES、HORZSIZE、VERTRES 和 VERTSIZE 计算出来的。这仅对于 Windows NT 下的视频显示器才有所不同。

EzCreateFont 函数开始只进行一些用于 Windows NT 的调整，即调用 SetGraphicsMode 和 ModifyWorldTransform 函数，它们在 Windows 98 下不起作用。因为 Windows NT 的全球转换应该有修改字体可视大小的作用，因此在计算字体大小之前，全球转换设置为默认值——无转换。

EzCreateFont 基本上设置 LOGFONT 结构的字段并调用 CreateFontIndirect，而 CreateFontIndirect 返回字体的句柄。EzCreateFont 函数的主要任务是将字体的磅值转换为 LOGFONT 结构的 lfHeight 字段所要求的逻辑单位。其实是首先将磅值转换为设备单位(像素)，然后再转换为逻辑单位。为完成第一步，函数使用 GetDeviceCaps。从像素到逻

辑单位的转换似乎只需简单地调用 DptoLP("从设备点到逻辑点")函数。但是为了使 DPtoLP 转换正常工作,在以后使用创建的字体显示文本时,相同的映射方式必须有效。这就意味着应该在调用 EzCreateFont 函数前设置映射方式。在大多数情况下,只使用一种映射方式在窗口的特定区域绘制,因此这种要求不是什么问题。

程序 17-3 所示的 EZTEST 程序不很严格地考验了 EZFONT 文件。此程序使用上面的 EZTEST 文件,还包括了本书后面程序要使用的 FONTDEMO 文件。

程序 17-3　EZTEST 程序

EZTEST.C

```
/*-------------------
    EZTEST.C——Test of EZFONT
              (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>
#include "ezfont.h"

TCHAR szAppName[] = TEXT("EZTest") ;
TCHAR szTitle     [] = TEXT("EZTest: Test of EZFONT") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     HFONT      hFont ;
     int        y, iPointSize ;
     LOGFONT    lf ;
     TCHAR      szBuffer[100] ;
     TEXTMETRIC tm ;

          // Set Logical Twips mapping mode

     SetMapMode (hdc, MM_ANISOTROPIC) ;
     SetWindowExtEx (hdc, 1440, 1440, NULL) ;
     SetViewportExtEx (hdc, GetDeviceCaps (hdc, LOGPIXELSX),
                            GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;

          // Try some fonts

     y = 0 ;

     for (iPointSize = 80 ; iPointSize <= 120 ; iPointSize++)
     {
          hFont = EzCreateFont (hdc, TEXT("Times New Roman"),
                                iPointSize, 0, 0, TRUE) ;

          GetObject (hFont, sizeof (LOGFONT), &lf) ;
```

```
            SelectObject (hdc, hFont) ;
            GetTextMetrics (hdc, &tm) ;
            TextOut (hdc, 0, y, szBuffer,
                wsprintf (szBuffer,
                        TEXT ("Times New Roman font of %i.%i points, ")
                        TEXT ("lf.lfHeight = %i, tm.tmHeight = %i"),
                        iPointSize / 10, iPointSize % 10,
                        lf.lfHeight, tm.tmHeight)) ;

            DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
            y += tm.tmHeight ;
        }
}
```

FONTDEMO.C

```
/*-----------------------
    FONTDEMO.C——Font Demonstration Shell Program
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>
#include "..\\EZTest\\EzFont.h"
#include "..\\EZTest\\resource.h"

extern void          PaintRoutine (HWND, HDC, int, int) ;
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

HINSTANCE hInst ;

extern TCHAR szAppName [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     TCHAR        szResource [] = TEXT ("FontDemo") ;
     HWND         hwnd ;
     MSG          msg ;
     WNDCLASS     wndclass ;

     hInst = hInstance ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
```

```
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = szResource ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, szTitle,
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;

}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static DOCINFO   di = { sizeof (DOCINFO), TEXT ("Font Demo: Printing") } ;
    static int       cxClient, cyClient ;
    static PRINTDLG  pd = { sizeof (PRINTDLG) } ;
    BOOL             fSuccess ;
    HDC              hdc, hdcPrn ;
    int              cxPage, cyPage ;
    PAINTSTRUCT      ps ;

    switch (message)
    {
    case WM_COMMAND:
        switch (wParam)
        {
```

```
case IDM_PRINT:

        // Get printer DC

    pd.hwndOwner = hwnd ;
    pd.Flags     = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;

    if (!PrintDlg (&pd))
        return 0 ;

    if (NULL == (hdcPrn = pd.hDC))
    {
         MessageBox (hwnd, TEXT ("Cannot obtain Printer DC"),
                     szAppName, MB_ICONEXCLAMATION | MB_OK) ;
         return 0 ;
    }
        // Get size of printable area of page

    cxPage = GetDeviceCaps (hdcPrn, HORZRES) ;
    cyPage = GetDeviceCaps (hdcPrn, VERTRES) ;

    fSuccess = FALSE ;

        // Do the printer page

    SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
    ShowCursor (TRUE) ;

    if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
    {
         PaintRoutine (hwnd, hdcPrn, cxPage, cyPage) ;

         if (EndPage (hdcPrn) > 0)
         {
              fSuccess = TRUE ;
              EndDoc (hdcPrn) ;
         }
    }
    DeleteDC (hdcPrn) ;

    ShowCursor (FALSE) ;
    SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

    if (!fSuccess)
        MessageBox (hwnd,
                    TEXT ("Error encountered during printing"),
                    szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;
```

```
            case IDM_ABOUT:
                MessageBox (hwnd, TEXT ("Font Demonstration Program \ n")
                                  TEXT ("(c) Charles Petzold, 1998"),
                            szAppName, MB_ICONINFORMATION | MB_OK) ;
                return 0 ;
            }
            break ;

        case WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            PaintRoutine (hwnd, hdc, cxClient, cyClient) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY :
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

FONTDEMO.RC

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

FONTDEMO MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Print...",                   IDM_PRINT
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About...",                   IDM_ABOUT
    END
END
```

```
RESOURCE.H
// Microsoft Developer Studio generated include file.
// Used by FontDemo.rc

#define IDM_PRINT                40001
#define IDM_ABOUT                40002
```

EZTEST.C 中的 PaintRoutine 函数将映射方式设置为 Logical Twips，然后创建字体范围为 8 磅～12 磅(间隔为 0.1 磅)的 Times New Roman 字体。第一次运行此程序时，它的输出可能会使您困惑。许多行文本使用大小明显相同的字体，并且 TEXTMETRIC 函数也报告这些字体具有相同的高度。这一切都是点阵过程的结果。显示器上的像素是不连续的，它不能显示每一个可能的字体大小。但是，FONTDEMO 外壳程序使打印输出的字体是不同的。这里您会发现字体大小区分得更加精确。

17.3.8 字体的旋转

您在 PICKFONT 中可能已经实验过了，LOGFONT 结构的 lfOrientation 和 lfEscapement 字段可以旋转 TrueType 文本。如果细致地考虑一下，这不会对 GDI 有很大的扩展。因为围绕原点旋转坐标点是大家都知道的。

虽然 EzCreateFont 不能指定字体的旋转角度，但是如 FONTROT("字体旋转")程序演示的那样，在调用函数后，进行调整是非常容易的。程序 17-4 显示了 FONTROT.C 文件，该程序也需要上面显示的 EZFONT 文件和 FONTDEMO 文件。

程序 17-4 FONTROT 程序

FONTROT.C

```
/*--------------------
    FONTROT.C——Rotated Fonts
                (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>
#include "..\\eztest\\ezfont.h"

TCHAR szAppName[] = TEXT ("FontRot") ;
TCHAR szTitle     [] = TEXT ("FontRot: Rotated Fonts") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     static TCHAR szString [] = TEXT (" Rotation") ;
     HFONT        hFont ;
     int          i ;
```

```
    LOGFONT    lf ;

    hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 540, 0, 0, TRUE) ;
    GetObject (hFont, sizeof (LOGFONT), &lf) ;
    DeleteObject (hFont) ;

    SetBkMode (hdc, TRANSPARENT) ;
    SetTextAlign (hdc, TA_BASELINE) ;
    SetViewportOrgEx (hdc, cxArea / 2, cyArea / 2, NULL) ;

    for (i = 0 ; i < 12 ; i++)
    {
         lf.lfEscapement = lf.lfOrientation = i * 300 ;
         SelectObject (hdc, CreateFontIndirect (&lf)) ;

         TextOut (hdc, 0, 0, szString, lstrlen (szString)) ;

         DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
    }
}
```

FONTROT 调用 EzCreateFont 仅为了获得与 54 磅 Times New Roman 字体相关的 LOGFONT 结构，然后程序删除该字体。在 for 循环中每隔 30°创建一个新字体，并显示文本，结果如图 17-2 所示。

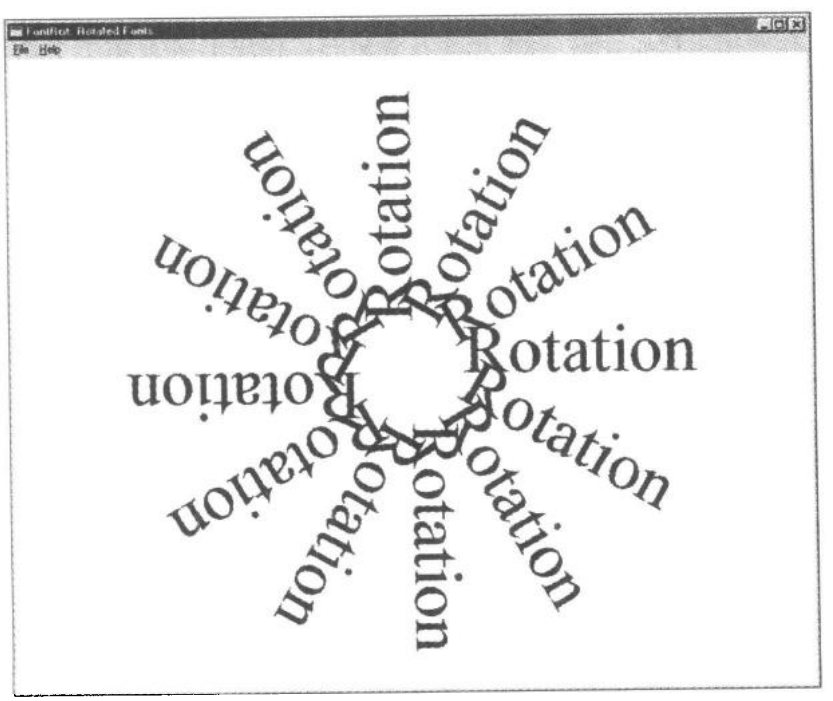


图 17-2　FONTROT 显示

如果对图形旋转和其他线性转换更加专业的方法感兴趣，并且知道您的程序在Windows NT下运行将受到限制，那么您可以使用XFORM矩阵和整体转换函数。

17.4　字体列举

字体列举是从GDI中获取设备的全部有效字体列表的过程。程序可以选择其中一种字体，或将它们显示在对话框中供用户选择。我先简单地介绍一下列举函数，然后显示使用ChooseFont函数的方法，ChooseFont降低了应用程序中进行字体列举的必要性。

17.4.1　列举函数

在Windows的早期，字体列举需要使用EnumFonts函数：

```
EnumFonts (hdc, szTypeFace, EnumProc, pData) ;
```

程序可以列举所有的字体(将第2个参数设置为NULL)或只列出特定的字样。第3个参数是列举回调函数；第4个参数是传递给该函数的可选数据。GDI为系统中的每种字体调用回调函数，将定义字体的LOGFONT和TEXTMETRIC结构以及一些表示字体类型的标志传递给它。

EnumFontFamilies函数是Windows 3.1下更好列举TrueType字体的函数：

```
EnumFontFamilies (hdc, szFaceName, EnumProc, pData) ;
```

通常第一次调用EnumFontFamilies时，第二个参数设置为NULL。为每个字体族(例如Times New Roman)调用一次EnumProc回调函数，然后应用程序使用该字体名和不同的回调函数再次调用EnumFontFamilies。GDI为字体族中的每种字体(例如Times New Roman Italic)调用第二个回调函数。对于非TrueType字体，向回调函数传递ENUMLOGFONT结构(它是由LOGFONT结构加上“全名”字段和“类型”字段构成，“类型”字段如文本名“Italic”或“Bold”)和TEXTMETRIC结构，对于TrueType字体传递NEWTEXTMETRIC结构。NEWTEXTMETRIC结构相对于TEXTMETRIC结构中的信息添加了4个字段。

EnumFontFamiliesEx函数推荐在Windows的32位的版本下使用：

```
EnumFontFamiliesEx (hdc, &logfont, EnumProc, pData, dwFlags) ;
```

第2个参数是指向LOGFONT结构的指针，其中lfCharSet和lfFaceName字段指出了所要列举的字体信息。回调函数在ENUMLOGFONTEX和NEWTEXTMETRICEX结构中得到每种字体的信息。

17.4.2　ChooseFont对话框

在第十一章简单介绍了ChooseFont通用对话框。现在，我们讨论字体列举，需要细致了

解一下 ChooseFont 函数的内部工作原理。ChooseFont 函数得到指向 CHOOSEFONT 结构的指针以此作为它的唯一参数，并显示列出所有字体的对话框。利用从 ChooseFont 中的返回值，LOGFONT 结构（CHOOSEFONT 结构的一部分）能够创建逻辑字体。

程序 17-5 所示的 CHOSFONT 程序演示了使用 ChooseFont 函数的方法，并显示了函数定义的 LOGFONT 结构的字段。程序也显示了在 PICKFONT 中显示的相同文本串。

程序 17-5　CHOSFONT 程序

CHOSFONT.C

```
/*--------------------
    CHOSFONT.C——ChooseFont Demo
                (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("ChosFont") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }
```

```
     hwnd = CreateWindow (szAppName, TEXT ("ChooseFont"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static CHOOSEFONT  cf ;
     static int         cyChar ;
     static LOGFONT     lf ;
     static TCHAR       szText[] = TEXT ("\x41\x42\x43\x44\x45")
                                   TEXT ("\x61\x62\x63\x64\x65")
                                   TEXT ("\xC0\xC1\xC2\xC3\xC4\xC5")
                                   TEXT ("\xE0\xE1\xE2\xE3\xE4\xE5")
# ifdef UNICODE
                                   TEXT ("\x0390\x0391\x0392\x0393\x0394\x0395")
                                   TEXT ("\x03B0\x03B1\x03B2\x03B3\x03B4\x03B5")

                                   TEXT ("\x0410\x0411\x0412\x0413\x0414\x0415")
                                   TEXT ("\x0430\x0431\x0432\x0433\x0434\x0435")

                                   TEXT ("\x5000\x5001\x5002\x5003\x5004")
# endif
                                   ;
     HDC                hdc ;
     int                y ;
     PAINTSTRUCT        ps ;
     TCHAR              szBuffer[64] ;
     TEXTMETRIC         tm ;

     switch (message)
     {
     case WM_CREATE:

          // Get text height

          cyChar = HIWORD (GetDialogBaseUnits ()) ;
```

```
            // Initialize the LOGFONT structure

        GetObject (GetStockObject (SYSTEM_FONT), sizeof (lf), &lf) ;

            // Initialize the CHOOSEFONT structure

        cf.lStructSize          = sizeof (CHOOSEFONT) ;
        cf.hwndOwner            = hwnd ;
        cf.hDC                  = NULL ;
        cf.lpLogFont            = &lf ;
        cf.iPointSize           = 0 ;
        cf.Flags                = CF_INITTOLOGFONTSTRUCT |
                                  CF_SCREENFONTS | CF_EFFECTS ;
        cf.rgbColors            = 0 ;
        cf.lCustData            = 0 ;
        cf.lpfnHook             = NULL ;
        cf.lpTemplateName       = NULL ;
        cf.hInstance            = NULL ;
        cf.lpszStyle            = NULL ;
        cf.nFontType            = 0 ;
        cf.nSizeMin             = 0 ;
        cf.nSizeMax             = 0 ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FONT:
            if (ChooseFont (&cf))
                InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

            // Display sample text using selected font

        SelectObject (hdc, CreateFontIndirect (&lf)) ;
        GetTextMetrics (hdc, &tm) ;
        SetTextColor (hdc, cf.rgbColors) ;
        TextOut (hdc, 0, y = tm.tmExternalLeading, szText, lstrlen (szText)) ;

            // Display LOGFONT structure fields using system font

        DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
```

```
SetTextColor (hdc, 0) ;

TextOut (hdc, 0, y += tm.tmHeight, szBuffer,
    wsprintf (szBuffer, TEXT ("lfHeight = %i"), lf.lfHeight)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfWidth = %i"), lf.lfWidth)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfEscapement = %i"),
              lf.lfEscapement)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfOrientation = %i"),
              lf.lfOrientation)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfWeight = %i"), lf.lfWeight)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfItalic = %i"), lf.lfItalic)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfUnderline = %i"), lf.lfUnderline)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfStrikeOut = %i"), lf.lfStrikeOut)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfCharSet = %i"), lf.lfCharSet)) ;
TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfOutPrecision = %i"),
              lf.lfOutPrecision)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfClipPrecision = %i"),
              lf.lfClipPrecision)) ;
TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfQuality = %i"), lf.lfQuality)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfPitchAndFamily = 0x%02X"),
              lf.lfPitchAndFamily)) ;

TextOut (hdc, 0, y += cyChar, szBuffer,
    wsprintf (szBuffer, TEXT ("lfFaceName = %s"), lf.lfFaceName)) ;

EndPaint (hwnd, &ps) ;
return 0 ;
```

```
    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

```
CHOSFONT.RC
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
CHOSFONT MENU DISCARDABLE
BEGIN
    MENUITEM "&Font!",                      IDM_FONT
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by ChosFont.rc

#define IDM_FONT                40001
```

与一般的对话框一样，CHOOSEFONT 结构的 Flags 字段列出了许多选项。CHOSFONT 指定的 CF_INITLOGFONTSTRUCT 标志使 Windows 根据传递给 ChooseFont 结构的 LOGFONT 结构对对话框的选择进行初始化。您可以使用标志来仅指定 TrueType 字体(CF_TTONLY)或仅指定等宽字体(CF_FIXEDPITCHONLY)或无符号字体(CF_SCRIPTSONLY)。也可以显示屏幕字体(CF_SCREENFONTS)、打印字体(CF_PRINTERFONTS)，或者两者都显示(CF_BOTH)。在后两种情况下，CHOOSEFONT 结构的 hDC 字段必须是打印机设备描述表。CHOSFONT 程序使用 CF_SCREENFONTS 标志。

CF_EFFECTS 标志(CHOSFONT 程序使用的第三个标志)强迫对话框包括用于下划线和删除线的复选框，并且允许选择文本的颜色。在代码中应用文本颜色不难，您可以试一试。

注意 Font 对话框中由 ChooseFont 显示的 Script 字段。它让用户选择用于特殊字体的字符集，适当的字符集 ID 在 LOGFONT 结构中返回。

ChooseFont 函数使用逻辑英寸从磅值中计算 lfHeight 字段。例如，假定您从“显示属性”对话框中设置了“小字体”。这意味着带有视频显示设备描述表的 GetDeviceCaps 和

参数 LOGPIXELSY 返回 96。如果使用 ChooseFont 选择 72 磅的 Times Roman 字体，实际上是想要 1 英寸高的字体。当 ChooseFont 返回后，LOGFONT 结构的 lfHeight 字段等于 -96（注意负号），这是指字体的磅值等于 96 像素，或者 1 逻辑英寸。

以上大概是我们想要知道的。但请记住以下几点：

- 如果在 Windows NT 下设置了度量映射方式，则逻辑坐标与字体的物理大小不一致。例如，如果在基于度量映射方式的文本旁画一把尺子，会发现它与字体不匹配。应该使用上面描述的 Logical Twips 映射方式来绘制图形，才能与字体大小一致。
- 如果要使用任何非 MM_TEXT 映射方式，请确保在把字体选入设备描述表和显示文本时，没有设置映射方式。否则，GDI 会认为 LOGFONT 结构的 lfHeight 字段是逻辑坐标。
- 由 ChooseFont 设置的 LOGFONT 结构的 lfHeight 字段总是像素值，并且它只适用于视频显示器。当您为打印机设备描述表创建字体时，必须调整 lfHeight 值。ChooseFont 函数使用 CHOOSEFONT 结构的 hDC 字段只为获得列在对话框中的打印机字体。此设备描述表句柄不影响 lfHeight 值。

幸运的是，CHOOSEFONT 结构包括一个 iPointSize 字段，它提供以 1/10 磅为单位的所选字体的大小。无论是什么设备描述表和映射方式，都能把这个字段转化为逻辑大小并用于 lfHeight 字段。在 EZFONT.C 中能找到合适的代码。可以根据您的需要简化它。

另一个使用 ChooseFont 的程序是 UNICHARS，程序 17-6 让您查看一种字体的所有字符，对于研究 Lucida Sans Unicode 字体（默认的显示字体）或 Bitstream CyberBit 字体尤其有用。UNICHARS 总是使用 TextOutW 函数来显示字体的字符，因此可以在 Windows NT 或 Windows 98 下运行它。

程序 17-6　UNICHARS 程序

UNICHARS.C

```
/*-----------------------
   UNICHARS.C——Displays 16-bit character codes
               (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
```

```
{
     static TCHAR  szAppName[ ] = TEXT ("UniChars") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requies Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Unicode Characters"),
                          WS_OVERLAPPEDWINDOW | WS_VSCROLL,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static CHOOSEFONT  cf ;
     static int         iPage ;
     static LOGFONT     lf ;
     HDC                hdc ;
     int                cxChar, cyChar, x, y, i, cxLabels ;
```

```
PAINTSTRUCT     ps ;
SIZE            size ;
TCHAR           szBuffer [8] ;
TEXTMETRIC      tm ;
WCHAR           ch ;

switch (message)
{
case WM_CREATE:
    hdc = GetDC (hwnd) ;
    lf.lfHeight = - GetDeviceCaps (hdc, LOGPIXELSY) / 6 ; // 12 points
    lstrcpy (lf.lfFaceName, TEXT ("Lucida Sans Unicode")) ;
    ReleaseDC (hwnd, hdc) ;

    cf.lStructSize  = sizeof (CHOOSEFONT) ;
    cf.hwndOwner = hwnd ;
    cf.lpLogFont  = &lf ;
    cf.Flags      = CF_INITTOLOGFONTSTRUCT | CF_SCREENFONTS ;

    SetScrollRange (hwnd, SB_VERT, 0, 255, FALSE) ;
    SetScrollPos   (hwnd, SB_VERT, iPage, TRUE ) ;
    return 0 ;

case WM_COMMAND:
    switch (LOWORD (wParam))
    {
    case IDM_FONT:
        if (ChooseFont (&cf))
            InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;
    }
    return 0 ;

case WM_VSCROLL:
    switch (LOWORD (wParam))
    {
        case SB_LINEUP:         iPage -= 1 ;      break ;
        case SB_LINEDOWN:       iPage += 1 ;      break ;
        case SB_PAGEUP:         iPage -= 16 ;     break ;
        case SB_PAGEDOWN:       iPage += 16 ;     break ;
        case SB_THUMBPOSITION:  iPage = HIWORD (wParam) ;   break ;
        default:
                return 0 ;
        }

    iPage = max (0, min (iPage, 255)) ;
    SetScrollPos (hwnd, SB_VERT, iPage, TRUE) ;
```

```
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            SelectObject (hdc, CreateFontIndirect (&lf)) ;

            GetTextMetrics (hdc, &tm) ;
            cxChar = tm.tmMaxCharWidth ;
            cyChar = tm.tmHeight + tm.tmExternalLeading ;

            cxLabels = 0 ;

            for (i = 0 ; i < 16 ; i++)
            {
                wsprintf (szBuffer, TEXT ("000%1X:"), i) ;
                GetTextExtentPoint (hdc, szBuffer, 7, &size) ;

                cxLabels = max (cxLabels, size.cx) ;
            }

            for (y = 0 ; y < 16 ; y++)
            {
                wsprintf (szBuffer, TEXT ("%03X:"), 16 * iPage + y) ;
                TextOut (hdc, 0, y * cyChar, szBuffer, 7) ;

                for (x = 0 ; x < 16 ; x++)
                {
                    ch = (WCHAR) (256 * iPage + 16 * y + x) ;
                    TextOutW (hdc, x * cxChar + cxLabels,
                                   y * cyChar, &ch, 1) ;
                }
            }

            DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

UNICHARS.RC

```
// Microsoft Developer Studio generated resource script.
```

```
# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

UNICHARS MENU DISCARDABLE
BEGIN
    MENUITEM "&Font!",                      IDM_FONT
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Unichars.rc

# define IDM_FONT                40001
```

17.5 段落格式化

具有选择并创建逻辑字体的能力后,就可以处理文本格式了。这个过程包括以 4 种方式之一来把文本的每一行放在页边距内:左对齐、右对齐、居中或分散对齐(从页边距的一端到另一端,字间距相等)。对于前三种任务,可以使用带有 DT_WORDBREAK 参数的 DrawText 函数,但这种方法有局限性。例如,您无法确定 DrawText 会把文本的哪个部分恰好放在矩形内。DrawText 对于一些简单任务是很方便的,但对更复杂的格式化任务,则可能要用到 TextOut。

17.5.1 简单文本格式化

对文本最有用的一个函数是 GetTextExtentPoint32。(这个函数的名称显示了 Windows 早期版本的一些变化。)该函数根据设备描述表中选入的当前字体得出字符串的宽度和高度:

```
GetTextExtentPoint32 (hdc, pString, iCount, &size) ;
```

逻辑单位的文本宽度和高度在 SIZE 结构的 cx 和 cy 字段中返回。我使用一行文本的例子,假定您把一种字体选入设备描述表,现在要写入文本:

```
TCHAR * szText [] = TEXT ("Hello, how are you?") ;
```

您希望文本从垂直坐标 yStart 开始,页边距由坐标 xLeft 和 xRight 设置。您的任务就是计算文本开始处的水平坐标的 xStart 值。

如果文本以等宽字体显示,那么这项任务就相当容易,但通常不是这样的。首先您

得到字符串的文本宽度：

```
GetTextExtentPoint32 (hdc, szText, lstrlen (szText), &size) ;
```

如果 size.cx 比(xRight - xLeft)大，这一行就太长了，不能放在页边距内。我们假定它能放进去。

要左对齐文本，只要把 xStart 设置为与 xLeft 相等，然后写入文本：

```
TextOut (hdc, xStart, yStart, szText, lstrlen (szText)) ;
```

这很容易。现在可以把 size.cy 加到 yStart 中写下一行文本了。

要右对齐文本，用以下公式计算 xStart：

```
xStart = xRight - size.cx ;
```

要居中文本用以下公式：

```
xStart = (xLeft + xRight - size.cx) / 2 ;
```

现在开始艰巨的任务——在左右页边距内分散对齐文本。页边距之间的距离是(xRight - xLeft)。如不调整，文本宽度就是 size.cx。两者之差

```
xRight - xLeft - size.cx
```

必须在字符串的三个空格字符处平均分配，这听起来很讨厌，但还不是太糟。可以调用

```
SetTextJustification (hdc, xRight - xLeft - size.cx, 3)
```

来完成。第二个参数是字符串内空格字符中需要分配的空间量。第三个参数是空格字符的数量，这里为 3。现在把 xStart 设置与 xLeft 相等，用 TextOut 写入文本：

```
TextOut (hdc, xStart, yStart, szText, lstrlen (szText)) ;
```

文本会在 xLeft 和 xRight 页边距之间分散对齐。

无论何时调用 SetTextJustification，如果空间量不能在空格字符中平均分配，它就会累积一个错误值。这将影响后面的 GetTextExtentPoint32 调用。每次开始新的一行，都必须通过调用

```
SetTextJustification (hdc, 0, 0) ;
```

来清除错误值。

17.5.2　使用段落

如果处理整个段落，就必须从头开始并扫描字符串来寻找空格字符。每当碰到一个空格(或其他能用于断行的字符)，需调用 GetTextExtentPoint32 来确定文本是否能放入左右页边距之间。当文本超出允许的空间时，就要退回上一个空白。现在，您已经能够确

定一行的字符串了。如果想要分散对齐该行，调用 SetText Justification 和 TextOut，清除错误值，并继续下一行。

显示在程序 17-7 中的 JUSTIFY1 对 Mark Twain 著《The Adventures of Huckleberry Finn》中的第一段做了这样的处理。您可以从对话框中选择想要的字体，也可以使用菜单选项来更改对齐方式(左对齐、右对齐、居中或分散对齐)。图 17-3 是典型的 JUSTIFY1 显示。

程序 17-7 JUSTIFY1 程序

JUSTIFY1.C

```
/* ---------------------------
    JUSTIFY1.C——Justified Type Program #1
                (c) Charles Petzold, 1998
   ---------------------------*/
#include <windows.h>
#include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName[] = TEXT ("Justify1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    HWND        hwnd ;
    MSG         msg ;
    WNDCLASS    wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = szAppName ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
            szAppName, MB_ICONERROR) ;
        return 0 ;
    }
```

```
     hwnd = CreateWindow (szAppName, TEXT ("Justified Type # 1"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

void DrawRuler (HDC hdc, RECT * prc)
{
     static int iRuleSize [16] = { 360, 72, 144, 72, 216, 72, 144, 72,
                                   288, 72, 144, 72, 216, 72, 144, 72 } ;
     int      i, j ;
     POINT    ptClient ;

     SaveDC (hdc) ;

          // Set Logical Twips mapping mode

     SetMapMode (hdc, MM_ANISOTROPIC) ;
     SetWindowExtEx (hdc, 1440, 1440, NULL) ;
     SetViewportExtEx (hdc, GetDeviceCaps (hdc, LOGPIXELSX),
                            GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;

          // Move the origin to a half inch from upper left

     SetWindowOrgEx (hdc, -720, -720, NULL) ;

          // Find the right margin (quarter inch from right)

     ptClient.x = prc->right ;
     ptClient.y = prc->bottom ;
     DPtoLP (hdc, &ptClient, 1) ;
     ptClient.x -= 360 ;

          // Draw the rulers

     MoveToEx (hdc, 0,          -360, NULL) ;
     LineTo    (hdc, ptClient.x, -360) ;
```

```
        MoveToEx (hdc, -360,          0, NULL) ;
        LineTo   (hdc, -360, ptClient.y) ;

        for (i = 0, j = 0 ; i <= ptClient.x ; i += 1440 / 16, j++)
        {
            MoveToEx (hdc, i, -360, NULL) ;
            LineTo   (hdc, i, -360 - iRuleSize [j % 16]) ;
        }

        for (i = 0, j = 0 ; i <= ptClient.y ; i += 1440 / 16, j++)
        {
            MoveToEx (hdc, -360, i, NULL) ;
            LineTo   (hdc, -360 - iRuleSize [j % 16], i) ;
        }

        RestoreDC (hdc, -1) ;
}

void Justify (HDC hdc, PTSTR pText, RECT * prc, int iAlign)
{
        int       xStart, yStart, cSpaceChars ;
        PTSTR     pBegin, pEnd ;
        SIZE      size ;

        yStart = prc->top ;
        do                                  // for each text line
        {
            cSpaceChars = 0 ;               // initialize number of spaces in line

            while (*pText == ' ')           // skip over leading spaces
                pText++ ;

            pBegin = pText ;                // set pointer to char at beginning of line

            do                              // until the line is known
            {
                pEnd = pText ;              // set pointer to char at end of line
                    // skip to next space

                while (*pText != '\0' && *pText++ != ' ') ;

                if (*pText == '\0')
                    break ;

                    // after each space encountered, calculate extents

                cSpaceChars++ ;
                GetTextExtentPoint32(hdc, pBegin, pText - pBegin - 1, &size) ;
            }
```

```
while (size.cx < (prc -> right - prc -> left)) ;

cSpaceChars - - ;                    // discount last space at end of line

while ( * (pEnd - 1) == ' ')    // eliminate trailing spaces
{
     pEnd - - ;
     cSpaceChars - - ;
}

    // if end of text and no space characters, set pEnd to end

if ( * pText == '\0' || cSpaceChars <= 0)
     pEnd = pText ;

GetTextExtentPoint32 (hdc, pBegin, pEnd - pBegin, &size) ;

switch (iAlign)                      // use alignment for xStart
{
case IDM_ALIGN_LEFT:
     xStart = prc -> left ;
     break ;

case IDM_ALIGN_RIGHT:
     xStart = prc -> right - size.cx ;
     break ;

case IDM_ALIGN_CENTER:
     xStart = (prc -> right + prc -> left - size.cx) / 2 ;
     break ;

case IDM_ALIGN_JUSTIFIED:
     if ( * pText != '\0' && cSpaceChars > 0)
          SetTextJustification (hdc,
                                prc -> right - prc -> left - size.cx,
                                cSpaceChars) ;
     xStart = prc -> left ;
     break ;
}
     // display the text

TextOut (hdc, xStart, yStart, pBegin, pEnd - pBegin) ;

     // prepare for next line

SetTextJustification (hdc, 0, 0) ;
yStart += size.cy ;
pText = pEnd ;
```

```
    }
    while ( * pText && yStart < prc -> bottom - size.cy) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static CHOOSEFONT  cf ;
    static DOCINFO     di = { sizeof (DOCINFO), TEXT ("Justify1: Printing") } ;
    static int         iAlign = IDM_ALIGN_LEFT ;
    static LOGFONT     lf ;
    static PRINTDLG    pd ;
    static TCHAR       szText[ ] = {
                           TEXT ("You don't know about me, without you ")
                           TEXT ("have read a book by the name of \"The ")
                           TEXT ("Adventures of Tom Sawyer, \" but that ")
                           TEXT ("ain't no matter. That book was made by ")
                           TEXT ("Mr. Mark Twain, and he told the truth, ")
                           TEXT ("mainly. There was things which he ")
                           TEXT ("stretched, but mainly he told the truth. ")
                           TEXT ("That is nothing. I never seen anybody ")
                           TEXT ("but lied, one time or another, without ")
                           TEXT ("it was Aunt Polly, or the widow, or ")
                           TEXT ("maybe Mary. Aunt Polly -- Tom's Aunt ")
                           TEXT ("Polly, she is -- and Mary, and the Widow ")
                           TEXT ("Douglas, is all told about in that book ")
                           TEXT (" -- which is mostly a true book; with ")
                           TEXT ("some stretchers, as I said before.") } ;
    BOOL               fSuccess ;
    HDC                hdc, hdcPrn ;
    HMENU              hMenu ;
    int                iSavePointSize ;
    PAINTSTRUCT        ps ;
    RECT               rect ;

    switch (message)
    {
    case WM_CREATE:
         // Initialize the CHOOSEFONT structure

        GetObject (GetStockObject (SYSTEM_FONT), sizeof (lf), &lf) ;

        cf.lStructSize    = sizeof (CHOOSEFONT) ;
        cf.hwndOwner      = hwnd ;
        cf.hDC            = NULL ;
        cf.lpLogFont      = &lf ;
        cf.iPointSize     = 0 ;
        cf.Flags          = CF_INITTOLOGFONTSTRUCT | CF_SCREENFONTS |
                            CF_EFFECTS ;
```

```
            cf.rgbColors        = 0 ;
            cf.lCustData        = 0 ;
            cf.lpfnHook         = NULL ;
            cf.lpTemplateName   = NULL ;
            cf.hInstance        = NULL ;
            cf.lpszStyle        = NULL ;
            cf.nFontType        = 0 ;
            cf.nSizeMin         = 0 ;
            cf.nSizeMax         = 0 ;

            return 0 ;

        case WM_COMMAND:
            hMenu = GetMenu (hwnd) ;

            switch (LOWORD (wParam))
            {
            case IDM_FILE_PRINT:
                                    // Get printer DC

                pd.lStructSize  = sizeof (PRINTDLG) ;
                pd.hwndOwner = hwnd ;
                pd.Flags       = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;

                if (!PrintDlg (&pd))
                    return 0 ;

                if (NULL == (hdcPrn = pd.hDC))
                {
                    MessageBox (hwnd, TEXT ("Cannot obtain Printer DC"),
                                szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                    return 0 ;
                }
                // Set margins of 1 inch

                rect.left   = GetDeviceCaps (hdcPrn, LOGPIXELSX) -
                              GetDeviceCaps (hdcPrn, PHYSICALOFFSETX) ;
                rect.top    = GetDeviceCaps (hdcPrn, LOGPIXELSY) -
                              GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;

                rect.right  = GetDeviceCaps (hdcPrn, PHYSICALWIDTH) -
                              GetDeviceCaps (hdcPrn, LOGPIXELSX) -
                              GetDeviceCaps (hdcPrn, PHYSICALOFFSETX) ;

                rect.bottom = GetDeviceCaps (hdcPrn, PHYSICALHEIGHT) -
                              GetDeviceCaps (hdcPrn, LOGPIXELSY) -
                              GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;
```

```
            // Display text on printer

        SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
        ShowCursor (TRUE) ;

        fSuccess = FALSE ;
        if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
        {
                // Select font using adjusted lfHeight

            iSavePointSize = lf.lfHeight ;
            lf.lfHeight = -(GetDeviceCaps (hdcPrn, LOGPIXELSY) *
                                cf.iPointSize) / 720 ;

            SelectObject (hdcPrn, CreateFontIndirect (&lf)) ;
            lf.lfHeight = iSavePointSize ;

                // Set text color

            SetTextColor (hdcPrn, cf.rgbColors) ;

                // Display text

            Justify (hdcPrn, szText, &rect, iAlign) ;

            if (EndPage (hdcPrn) > 0)
            {
                fSuccess = TRUE ;
                EndDoc (hdcPrn) ;
            }
        }
        ShowCursor (FALSE) ;
        SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

        DeleteDC (hdcPrn) ;

        if (!fSuccess)
            MessageBox (hwnd, TEXT ("Could not print text"),
                        szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return 0 ;

    case IDM_FONT:
        if (ChooseFont (&cf))
            InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

    case IDM_ALIGN_LEFT:
    case IDM_ALIGN_RIGHT:
```

```
            case IDM_ALIGN_CENTER:
            case IDM_ALIGN_JUSTIFIED:
                CheckMenuItem (hMenu, iAlign, MF_UNCHECKED) ;
                iAlign = LOWORD (wParam) ;
                CheckMenuItem (hMenu, iAlign, MF_CHECKED) ;
                InvalidateRect (hwnd, NULL, TRUE) ;
                return 0 ;
            }
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;
            DrawRuler (hdc, &rect) ;

            rect.left   += GetDeviceCaps (hdc, LOGPIXELSX) / 2 ;
            rect.top    += GetDeviceCaps (hdc, LOGPIXELSY) / 2 ;
            rect.right  -= GetDeviceCaps (hdc, LOGPIXELSX) / 4 ;

            SelectObject (hdc, CreateFontIndirect (&lf)) ;
            SetTextColor (hdc, cf.rgbColors) ;

            Justify (hdc, szText, &rect, iAlign) ;

            DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

JUSTIFY1.RC

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

JUSTIFY1 MENU DISCARDABLE
BEGIN
```

```
    POPUP "&File"
    BEGIN
        MENUITEM "&Print",              IDM_FILE_PRINT
    END
    POPUP "&Font"
    BEGIN
        MENUITEM "&Font...",            IDM_FONT
    END
    POPUP "&Align"
    BEGIN
        MENUITEM "&Left",               IDM_ALIGN_LEFT, CHECKED
        MENUITEM "&Right",              IDM_ALIGN_RIGHT
        MENUITEM "&Centered",           IDM_ALIGN_CENTER
        MENUITEM "&Justified",          IDM_ALIGN_JUSTIFIED
    END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Justify1.rc

#define IDM_FILE_PRINT          40001
#define IDM_FONT                40002
#define IDM_ALIGN_LEFT          40003
#define IDM_ALIGN_RIGHT         40004
#define IDM_ALIGN_CENTER        40005
#define IDM_ALIGN_JUSTIFIED     40006
```

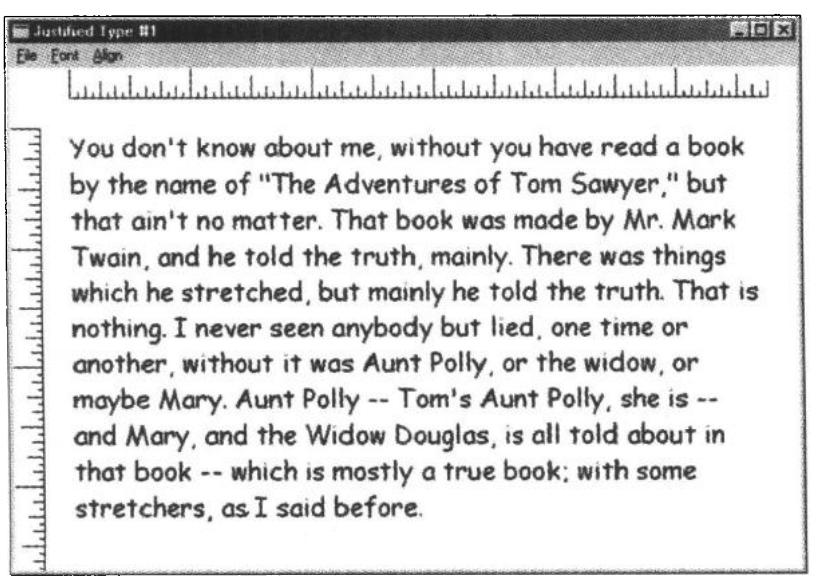

Justified Type #1
File Font Align
You don't know about me, without you have read a book by the name of "The Adventures of Tom Sawyer," but that ain't no matter. That book was made by Mr. Mark Twain, and he told the truth, mainly. There was things which he stretched, but mainly he told the truth. That is nothing. I never seen anybody but lied, one time or another, without it was Aunt Polly, or the widow, or maybe Mary. Aunt Polly -- Tom's Aunt Polly, she is -- and Mary, and the Widow Douglas, is all told about in that book -- which is mostly a true book; with some stretchers, as I said before.

图 17-3　典型的 JUSTIFY1 显示

JUSTIFY1 在客户区的上部和左侧显示了标尺(当然单位是逻辑英寸)。标尺由 DrawRuler 函数画出。一个矩形结构定义了分散对齐文本的区域。

涉及对文本进行格式化的大量工作由 Justify 函数实现。函数搜寻文本开始的空白,并使用 GetTextExtentPoint32 测量每一行。当行的长度超过显示区域的宽度时,JUSTIFY1 返回先前的空格,并使该行到达换行处。根据 iAlign 常量的值,行的对齐方式有:左对齐、右对齐、居中或分散对齐。

JUSTIFY1 并不完美。例如,它没有使用连字符的任何逻辑。此外,当每行少于两个字时,分散对齐逻辑会崩溃。即使我们解决了这个不是特别难的问题,当一个单词太长,在左右边距间放不下时,程序仍不能正常运作。当然,当我们在程序中对同一行使用多种字体(如同 Windows 字处理程序轻松做出的那样)时,情况会更复杂。还没有人声称这种处理容易,它只是比我们亲自做所有的工作容易一些。

17.5.3　打印输出预览

一些文本不只是用于在屏幕上查看的。一些文本是用于打印的。经常在这种情况下,文本的屏幕预览必须与打印输出的格式精确匹配。显示同样的字体、大小和字符格式是不够的。使用 TrueType 是个捷径。另外还需要将段落中的每一行在同样位置断开。这是 WYSIWYG 中的难点。

JUSTIFY1 包含一个"Print"选项,但该选项仅在页面的上、左和右边设置 1 英寸的边距。这样,格式化完全与屏幕显示器无关。这里有一个有趣的练习:在 JUSTIFY1 中更改几行代码,使屏幕和打印机逻辑基于一个 6 英寸的格式化矩形。方法就是在 WM_PAINT 和"Print"命令逻辑中更改 rect.right 的定义。在 WM_PAINT 逻辑中,相应语句为:

```
rect.right = rect.left + 6 * GetDeviceCaps (hdc, LOGPIXELSX) ;
```

在"Print"命令逻辑中,相应语句为:

```
rect.right = rect.left + 6 * GetDeviceCaps (hdcPrn, LOGPIXELSX) ;
```

如果选择了一种 TrueType 字体,屏幕上的换行情况应与打印机的输出相同。

但实际情况并不是这样。即使两种设备使用同样磅值的相同字体,并将文本显示在同样的格式化矩形中,不同的显示分辨率及凑整误差也会使换行出现在不同地方。显然,需要一种更高明的方法进行屏幕上的打印输出预览。

程序 17-8 所示的 JUSTIFY2 示范了这种方法的一个尝试。JUSTIFY2 中的代码是基于 Microsoft 的 David Weise 所写的 TTJUST("TrueType Justify")程序,而该程序又是基于本书前面的一个版本中的 JUSTIFY1 程序。为表现出这一程序中增加的复杂性,用 Herman Melville 著《Moby-Dick》的第一章代替了 Mark Twain 小说的摘录。

程序 17-8 JUSTIFY2 程序

JUSTIFY2.C

```
/*----------------------
   JUSTIFY2.C——Justified Type Program #2
               (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>
#include "resource.h"

#define OUTWIDTH 6          // Width of formatted output in inches
#define LASTCHAR 127        // Last character code used in text

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("Justify2") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;
          if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Justified Type #2"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

void DrawRuler (HDC hdc, RECT * prc)
{
     static int  iRuleSize [16] = { 360, 72, 144, 72, 216, 72, 144, 72,
                                    288, 72, 144, 72, 216, 72, 144, 72 } ;
     int         i, j ;
     POINT       ptClient ;

     SaveDC (hdc) ;

          // Set Logical Twips mapping mode

     SetMapMode (hdc, MM_ANISOTROPIC) ;
     SetWindowExtEx (hdc, 1440, 1440, NULL) ;
     SetViewportExtEx (hdc, GetDeviceCaps (hdc, LOGPIXELSX),
                       GetDeviceCaps (hdc, LOGPIXELSY), NULL) ;

          // Move the origin to a half inch from upper left

     SetWindowOrgEx (hdc, -720, -720, NULL) ;

          // Find the right margin (quarter inch from right)

     ptClient.x = prc->right ;
     ptClient.y = prc->bottom ;
     DPtoLP (hdc, &ptClient, 1) ;
     ptClient.x -= 360 ;

          // Draw the rulers

     MoveToEx  (hdc, 0,                    -360, NULL) ;
     LineTo    (hdc, OUTWIDTH * 1440,      -360) ;
     MoveToEx  (hdc, -360,                    0, NULL) ;
     LineTo    (hdc, -360,          ptClient.y) ;

     for (i = 0, j = 0 ; i <= ptClient.x && i <= OUTWIDTH * 1440 ;
```

```
                    i += 1440 / 16, j++)
    {
        MoveToEx (hdc, i, -360, NULL) ;
        LineTo   (hdc, i, -360 - iRuleSize [j % 16]) ;
    }

    for (i = 0, j = 0 ; i <= ptClient.y ; i += 1440 / 16, j++)
    {
        MoveToEx (hdc, -360, i, NULL) ;
        LineTo   (hdc, -360 - iRuleSize [j % 16], i) ;
    }

    RestoreDC (hdc, -1) ;
}

/*-------------------------------------------
    GetCharDesignWidths: Gets character widths for font as large as the
                         original design size
   -------------------------------------------*/

UINT GetCharDesignWidths (HDC hdc, UINT uFirst, UINT uLast, int * piWidths)
{
    HFONT             hFont, hFontDesign ;
    LOGFONT           lf ;
    OUTLINETEXTMETRIC otm ;

    hFont = GetCurrentObject (hdc, OBJ_FONT) ;
    GetObject (hFont, sizeof (LOGFONT), &lf) ;

        // Get outline text metrics (we'll only be using a field that is
        //     independent of the DC the font is selected into)

    otm.otmSize = sizeof (OUTLINETEXTMETRIC) ;
    GetOutlineTextMetrics (hdc, sizeof (OUTLINETEXTMETRIC), &otm) ;

        // Create a new font based on the design size

    lf.lfHeight   = - (int) otm.otmEMSquare ;
    lf.lfWidth    = 0 ;
    hFontDesign = CreateFontIndirect (&lf) ;

        // Select the font into the DC and get the character widths

    SaveDC (hdc) ;
    SetMapMode (hdc, MM_TEXT) ;
    SelectObject (hdc, hFontDesign) ;

    GetCharWidth (hdc, uFirst, uLast, piWidths) ;
```

```
    SelectObject (hdc, hFont) ;
    RestoreDC (hdc, -1) ;

        // Clean up

    DeleteObject (hFontDesign) ;

    return otm.otmEMSquare ;
}

/*----------------------------------
    GetScaledWidths: Gets floating point character widths for selected
                     font size
   ----------------------------------*/

void GetScaledWidths (HDC hdc, double * pdWidths)
{
    double      dScale ;
    HFONT       hFont ;
    int         aiDesignWidths [LASTCHAR + 1] ;
    int         i ;
    LOGFONT     lf ;
    UINT        uEMSquare ;

        // Call function above

    uEMSquare = GetCharDesignWidths (hdc, 0, LASTCHAR, aiDesignWidths) ;

        // Get LOGFONT for current font in device context

    hFont = GetCurrentObject (hdc, OBJ_FONT) ;
    GetObject (hFont, sizeof (LOGFONT), &lf) ;
        // Scale the widths and store as floating point values

    dScale = (double) -lf.lfHeight / (double) uEMSquare ;

    for (i = 0 ; i <= LASTCHAR ; i++ )
        pdWidths[i] = dScale * aiDesignWidths[i] ;
}

/*----------------------------
    GetTextExtentFloat: Calculates text width in floating point
   ----------------------------*/

double GetTextExtentFloat (double * pdWidths, PTSTR psText, int iCount)
{
    double  dWidth = 0 ;
    int     i ;
```

```
    for (i = 0 ; i < iCount ; i++)
        dWidth += pdWidths [psText[i]] ;

    return dWidth ;
}

/* - - - - - - - - - - - - - - - - - - - - - - - - -
    Justify: Based on design units for screen/printer compatibility
   - - - - - - - - - - - - - - - - - - - - - - - - - */

void Justify (HDC hdc, PTSTR pText, RECT * prc, int iAlign)
{
    double  dWidth, adWidths[LASTCHAR + 1] ;
    int     xStart, yStart, cSpaceChars ;
    PTSTR   pBegin, pEnd ;
    SIZE    size ;

        // Fill the adWidths array with floating point character widths

    GetScaledWidths (hdc, adWidths) ;

    yStart = prc -> top ;
    do                                      // for each text line
    {
        cSpaceChars = 0 ;                   // initialize number of spaces in line

        while ( * pText == ' ')             // skip over leading spaces
            pText ++ ;

        pBegin = pText ;                    // set pointer to char at beginning of line

        do                                  // until the line is known
        {
            pEnd = pText ;                  // set pointer to char at end of line

                // skip to next space

            while ( * pText != '\0' && * pText ++ != ' ') ;

            if ( * pText == '\0')
                break ;

                // after each space encountered, calculate extents

            cSpaceChars ++ ;
            dWidth = GetTextExtentFloat (adWidths, pBegin,
                                         pText - pBegin - 1) ;
        }
        while (dwidth < double) (prc -> right - pc -> left)) ;
```

```
cSpaceChars--;                          // discount last space at end of line

while (*(pEnd - 1) == ' ')              // eliminate trailing spaces
{
     pEnd--;
     cSpaceChars--;
}

    // if end of text and no space characters, set pEnd to end

if (*pText == '\0' || cSpaceChars <= 0)
     pEnd = pText;

     // Now get integer extents

GetTextExtentPoint32(hdc, pBegin, pEnd - pBegin, &size);

switch (iAlign)                         // use alignment for xStart
{
case IDM_ALIGN_LEFT:
    xStart = prc->left;
    break;

case IDM_ALIGN_RIGHT:
    xStart = prc->right - size.cx;
    break;

case IDM_ALIGN_CENTER:
    xStart = (prc->right + prc->left - size.cx) / 2;
    break;

case IDM_ALIGN_JUSTIFIED:
    if (*pText != '\0' && cSpaceChars > 0)
        SetTextJustification (hdc,
                              prc->right - prc->left - size.cx,
                              cSpaceChars);
    xStart = prc->left;
    break;
}
     // display the text

TextOut (hdc, xStart, yStart, pBegin, pEnd - pBegin);

     // prepare for next line

SetTextJustification (hdc, 0, 0);
yStart += size.cy;
```

```
            pText = pEnd ;
        }
        while ( * pText && yStart < prc -> bottom - size.cy) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        static CHOOSEFONT  cf ;
        static DOCINFO     di = { sizeof (DOCINFO), TEXT ("Justify2: Printing") } ;
        static int         iAlign = IDM_ALIGN_LEFT ;
        static LOGFONT     lf ;
        static PRINTDLG    pd ;
        static TCHAR       szText[] = {
                                TEXT ("Call me Ishmael. Some years ago——never ")
                                TEXT ("mind how long precisely——having little ")
                                TEXT ("or no money in my purse, and nothing ")
                                TEXT ("particular to interest me on shore, I ")
                                TEXT ("thought I would sail about a little and ")
                                TEXT ("see the watery part of the world. It is ")
                                TEXT ("a way I have of driving off the spleen, ")
                                TEXT ("and regulating the circulation. Whenever ")
                                TEXT ("I find myself growing grim about the ")
                                TEXT ("mouth; whenever it is a damp, drizzly ")
                                TEXT ("November in my soul; whenever I find ")
                                TEXT ("myself involuntarily pausing before ")
                                TEXT ("coffin warehouses, and bringing up the ")
                                TEXT ("rear of every funeral I meet; and ")
                                TEXT ("especially whenever my hypos get such an ")
                                TEXT ("upper hand of me, that it requires a ")
                                TEXT ("strong moral principle to prevent me ")
                                TEXT ("from deliberately stepping into the ")
                                TEXT ("street, and methodically knocking ")
                                TEXT ("people's hats off——then, I account it ")
                                TEXT ("high time to get to sea as soon as I ")
                                TEXT ("can. This is my substitute for pistol ")
                                TEXT ("and ball. With a philosophical flourish ")
                                TEXT ("Cato throws himself upon his sword; I ")
                                TEXT ("quietly take to the ship. There is ")
                                TEXT ("nothing surprising in this. If they but ")
                                TEXT ("knew it, almost all men in their degree, ")
                                TEXT ("some time or other, cherish very nearly ")
                                TEXT ("the same feelings towards the ocean with ")
                                TEXT ("me.") } ;
        BOOL               fSuccess ;
        HDC                hdc, hdcPrn ;
        HMENU              hMenu ;
        int                iSavePointSize ;
        PAINTSTRUCT        ps ;
```

```
    RECT            rect ;

    switch (message)
    {
    case WM_CREATE:
            // Initialize the CHOOSEFONT structure

        hdc = GetDC (hwnd) ;
        lf.lfHeight = - GetDeviceCaps (hdc, LOGPIXELSY) / 6 ;
        lf.lfOutPrecision = OUT_TT_ONLY_PRECIS ;
        lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;
        ReleaseDC (hwnd, hdc) ;

        cf.lStructSize    = sizeof (CHOOSEFONT) ;
        cf.hwndOwner      = hwnd ;
        cf.hDC            = NULL ;
        cf.lpLogFont      = &lf ;
        cf.iPointSize     = 120 ;

            // Set flags for TrueType only!

        cf.Flags          = CF_INITTOLOGFONTSTRUCT | CF_SCREENFONTS |
                            CF_TTONLY | CF_EFFECTS ;
        cf.rgbColors      = 0 ;
        cf.lCustData      = 0 ;
        cf.lpfnHook       = NULL ;
        cf.lpTemplateName = NULL ;
        cf.hInstance      = NULL ;
        cf.lpszStyle      = NULL ;
        cf.nFontType      = 0 ;
        cf.nSizeMin       = 0 ;
        cf.nSizeMax       = 0 ;

        return 0 ;

    case WM_COMMAND:
        hMenu = GetMenu (hwnd) ;

        switch (LOWORD (wParam))
        {
        case IDM_FILE_PRINT:
                // Get printer DC

            pd.lStructSize  = sizeof (PRINTDLG) ;
            pd.hwndOwner = hwnd ;
            pd.Flags        = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;
            if (!PrintDlg (&pd))
                return 0 ;
```

```
if (NULL == (hdcPrn = pd.hDC))
{
     MessageBox (hwnd, TEXT ("Cannot obtain Printer DC"),
                 szAppName, MB_ICONEXCLAMATION | MB_OK) ;
     return 0 ;
}
    // Set margins for OUTWIDTH inches wide

rect.left      = (GetDeviceCaps (hdcPrn, PHYSICALWIDTH) -
                  GetDeviceCaps (hdcPrn, LOGPIXELSX) * OUTWIDTH) / 2
                - GetDeviceCaps (hdcPrn, PHYSICALOFFSETX) ;

rect.right     = rect.left +
                 GetDeviceCaps (hdcPrn, LOGPIXELSX) * OUTWIDTH ;

    // Set margins of 1 inch at top and bottom

rect.top       = GetDeviceCaps (hdcPrn, LOGPIXELSY) -
                 GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;

rect.bottom    = GetDeviceCaps (hdcPrn, PHYSICALHEIGHT) -
                 GetDeviceCaps (hdcPrn, LOGPIXELSY) -
                 GetDeviceCaps (hdcPrn, PHYSICALOFFSETY) ;

    // Display text on printer

SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
ShowCursor (TRUE) ;

fSuccess = FALSE ;

if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
{
          // Select font using adjusted lfHeight

     iSavePointSize = lf.lfHeight ;
     lf.lfHeight = -(GetDeviceCaps (hdcPrn, LOGPIXELSY) *
                          cf.iPointSize) / 720 ;

     SelectObject (hdcPrn, CreateFontIndirect (&lf)) ;
     lf.lfHeight = iSavePointSize ;

          // Set text color

     SetTextColor (hdcPrn, cf.rgbColors) ;

          // Display text
```

```
                Justify (hdcPrn, szText, &rect, iAlign) ;

                if (EndPage (hdcPrn) > 0)
                {
                    fSuccess = TRUE ;
                    EndDoc (hdcPrn) ;
                }
            }
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            DeleteDC (hdcPrn) ;

            if (!fSuccess)
                MessageBox (hwnd, TEXT ("Could not print text"),
                            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
            return 0 ;

        case IDM_FONT:
            if (ChooseFont (&cf))
                InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case IDM_ALIGN_LEFT:
        case IDM_ALIGN_RIGHT:
        case IDM_ALIGN_CENTER:
        case IDM_ALIGN_JUSTIFIED:
            CheckMenuItem (hMenu, iAlign, MF_UNCHECKED) ;
            iAlign = LOWORD (wParam) ;
            CheckMenuItem (hMenu, iAlign, MF_CHECKED) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;
        }
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;
        DrawRuler (hdc, &rect) ;

        rect.left   += GetDeviceCaps (hdc, LOGPIXELSX) / 2 ;
        rect.top    += GetDeviceCaps (hdc, LOGPIXELSY) / 2 ;
        rect.right = rect.left + OUTWIDTH * GetDeviceCaps (hdc, LOGPIXELSX) ;

        SelectObject (hdc, CreateFontIndirect (&lf)) ;
        SetTextColor (hdc, cf.rgbColors) ;
```

```
            Justify (hdc, szText, &rect, iAlign) ;

            DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

JUSTIFY2.RC

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

JUSTIFY2 MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Print",                  IDM_FILE_PRINT
    END
    POPUP "&Font"
    BEGIN
        MENUITEM "&Font...",                IDM_FONT
    END
    POPUP "&Align"
    BEGIN
        MENUITEM "&Left",                   IDM_ALIGN_LEFT, CHECKED
        MENUITEM "&Right",                  IDM_ALIGN_RIGHT
        MENUITEM "&Centered",               IDM_ALIGN_CENTER
        MENUITEM "&Justified",              IDM_ALIGN_JUSTIFIED
    END
END
```

RESOURCE.H

```
// Microsoft Developer Studio generated include file.
// Used by Justify2.rc

# define IDM_FILE_PRINT                     40001
```

```
# define IDM_FONT                40002
# define IDM_ALIGN_LEFT          40003
# define IDM_ALIGN_RIGHT         40004
# define IDM_ALIGN_CENTER        40005
# define IDM_ALIGN_JUSTIFIED     40006
```

JUSTIFY2 仅使用 TrueType 字体。在它的 GetCharDesignWidths 函数中，程序使用 GetOutlineTextMetrics 函数获取一条表面上似乎不重要的信息，即 OUTLINETEXTMETRIC 的 otmEMSquare 字段。

TrueType 字体在全方(em-square)的网格上设计(如我说过，“em”指一种方块类型的宽度，M 在宽度上等于字体磅值的大小)。任何特定 TrueType 字体的所有字符都是在同样的网格上设计的，虽然这些字符通常有不同的宽度。OUTLINETEXTMETRIC 结构的 otmEMSquare 字段给出了任意特定字体的这种全方形式的大小。您会发现：对于大多数 TrueType 字体，otmEMSquare 字段等于 2048，这意味着字体是在 2048 × 2048 的网格上设计的。

关键在于：可以为想要使用的特定 TrueType 字样名设置一个 LOGFONT 结构，其 lfHeight 字段等于 otmEMSquare 值的负数。在创建字体并将其选入设备描述表后，可调用 GetCharWidth。该函数以逻辑单位提供字体中单个字符的宽度。通常，因为这些字符被缩放为不同的字体大小，所以字符宽度并不准确。但使用基于 otmEMSquare 大小的字体，这些宽度总是与任何设备描述表无关的精确整数。

GetCharDesignWidths 函数以这种方式获得原始的字符设计宽度，并将它们存储在整数数组中。JUSTIFY2 程序在自己的文本中仅使用 ASCII 字符，因此，这个数组不需要很大。GetScaledWidths 函数将这些整型宽度转变为基于设备逻辑坐标中字体的实际磅值的浮点宽度。GetTextExtentFloat 函数使用这些浮点宽度计算整个字符串的宽度。这是新的 Justify 函数用以计算文本行宽度的操作。

17.6　娱乐性素材

根据外形轮廓表示字体字符提供了将字体与其他图形技术相结合的可能性。前面我们讨论了旋转字体的方式。这里讲述一些其他技巧。继续之前，先了解两个重要的预备知识：图形轨迹和扩展画笔。

17.6.1　GDI 轨迹

轨迹是存储在 GDI 内的直线和曲线的集合。轨迹是在 Windows 的 32 位版本中引入的。轨迹看上去类似于区域，我们确实可以将轨迹转换为区域，并使用轨迹进行剪取。但随后我们会发现两者的不同。

要定义轨迹,可先简单调用

```
BeginPath (hdc) ;
```

进行该调用之后,所画的任何线(例如,直线、弧及贝塞尔样条)将作为轨迹存储在 GDI 内部,不被提交到设备描述表上。轨迹经常由连接起来的线组成。要制作连接线,应使用 LineTo、PolylineTo 和 BezierTo 函数,这些函数都以当前位置为起点划线。如果使用 MoveToEx 改变了当前位置,或者调用其他的画线函数,或者调用了会导致当前位置改变的窗口/视口函数,您就在整个轨迹中创建了一个新的子轨迹。因此,轨迹包含一个或多个子轨迹,每一个子轨迹就是一系列连接的线段。

轨迹中的每个子轨迹可以是敞开的,也可以是封闭的。封闭子轨迹的第一条连接线的第一个点与最后一条连接线的最后一点相同,并且子轨迹通过调用 CloseFigure 结束。如果必要,CloseFigure 将用一条直线封闭子轨迹。随后的画线函数将开始一个新的子轨迹。最后,通过下面的调用结束轨迹定义:

```
EndPath (hdc) ;
```

这时,接着调用下列 5 个函数之一:

```
StrokePath (hdc) ;
FillPath (hdc) ;
StrokeAndFillPath (hdc) ;
hRgn = PathToRegion (hdc) ;
SelectClipPath (hdc, iCombine) ;
```

这些函数中的每一个都会在轨迹定义完成后,将其清除。

StrokePath 使用当前画笔绘制轨迹。您可能会好奇:点是什么?为什么不能跳过这些轨迹素材正常地画线?稍后我会告诉原因。

另外 4 个函数用直线关闭任何敞开的轨迹。FillPath 依照当前的多边填充模式使用当前画刷填充轨迹。StrokeAndFillPath 一次完成这两项工作。也可将轨迹转换为区域,或者将轨迹用于某个剪取区域。iCombine 参数是 CombineRgn 函数使用的 RGN_系列常量之一,它指出了轨迹与当前剪取区域的结合方式。

用于填充或剪取时,轨迹比区域更灵活,这是因为区域仅能由矩形、椭圆及多边形的组合定义,轨迹可由贝塞尔样条,至少在 Windows NT 中还可由弧线组成。在 GDI 中,轨迹和区域的存储也完全不同。轨迹是直线及曲线定义的集合,而区域(通常意义上)是扫描线的集合。

17.6.2 扩展画笔

在调用 StrokePath 时,使用当前画笔提交轨迹。在第四章讨论了用以创建画笔对象的 CreatePen 函数。伴随轨迹的引入,Windows 也支持一个被称为 ExtCreatePen 的扩展画笔

函数调用。该函数揭示了其创建轨迹以及使用轨迹要比不使用轨迹画线有用。ExtCreatePen 函数如下所示：

```
hPen = ExtCreatePen (iStyle, iWidth, &iBrush, 0, NULL) ;
```

您可以使用该函数正常地绘制线段，但在这种情况下某些特征不被 Windows 98 所支持。甚至在用以提交轨迹时，某些特征仍不被 Windows 98 所支持，这就是上面函数的最后两个参数被设置为 0 及 NULL 的原因。

对于 ExtCreatePen 的第一个参数，可使用第四章中所讨论的用在 CreatePen 上的所有样式。可使用 PS_GEOMETRIC 另外组合这些样式(其中 iWidth 参数以逻辑单位表示线宽，并能够转换)，或者使用 PS_COSMETIC(其中 iWidth 参数必须是 1)。在 Windows 98 中，虚线或点画线样式的画笔必须是 PS_COSMETIC，在 Windows NT 中取消了这个限制。

CreatePen 的一个参数表示颜色；ExtCreatePen 的相应参数不只表示颜色，它还使用画刷给 PS_GEOMETRIC 画笔内部着色。该画刷甚至能通过位图定义。

在绘制宽线段时，我们可能要关注线段端点的外观。在连接直线或曲线时，可能还要关注线段间连接点的外观。画笔由 CreatePen 创建时，这些端点及连接点通常是圆形的；使用 ExtCreatePen 创建画笔时我们可以选择。(实际上，在 Windows 98 中，仅当使用画笔实现轨迹时我们可以选择；在 Windows NT 中要更加灵活。)宽线段的端点可以使用 ExtCreatePen 中的下列画笔样式定义：

```
PS_ENDCAP_ROUND
PS_ENDCAP_SQUARE
PS_ENDCAP_FLAT
```

“square”样式与“flat”样式的不同点是：前者将线伸展到一半宽。与端点类似，轨迹中线段间的连接点可通过如下样式设置：

```
PS_JOIN_ROUND
PS_JOIN_BEVEL
PS_JOIN_MITER
```

“bevel”样式将连接点切断；“miter”样式将连接点变为箭头。程序 17-9 所示的 ENDJOIN 是对此的一个较好的说明。

程序 17-9　ENDJOIN 程序

ENDJOIN.C

```
/*-----------------------
   ENDJOIN.C——Ends and Joins Demo
              (c) Charles Petzold, 1998
   -----------------------*/
```

```
# include < windows.h >
LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[ ] = TEXT ("EndJoin") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Ends and Joins Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}
```

```
LRESULT CALLBACK WndProc (HWND hwnd, UINT iMsg, WPARAM wParam, LPARAM lParam)
{
     static int       iEnd[] = { PS_ENDCAP_ROUND, PS_ENDCAP_SQUARE, PS_ENDCAP_FLAT } ;
     static int       iJoin[] = { PS_JOIN_ROUND,   PS_JOIN_BEVEL,     PS_JOIN_MITER } ;
     static int       cxClient, cyClient ;
     HDC              hdc ;
     int              i ;
     LOGBRUSH         lb ;
     PAINTSTRUCT      ps ;

     switch (iMsg)
     {
     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_PAINT:
          hdc = BeginPaint (hwnd, &ps) ;

          SetMapMode (hdc, MM_ANISOTROPIC) ;
          SetWindowExtEx (hdc, 100, 100, NULL) ;
          SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;

          lb.lbStyle = BS_SOLID ;
          lb.lbColor = RGB (128, 128, 128) ;
          lb.lbHatch = 0 ;

          for (i = 0 ; i < 3 ; i++)
          {
               SelectObject (hdc,
                    ExtCreatePen (PS_SOLID | PS_GEOMETRIC |
                                  iEnd [i] | iJoin [i], 10,
                                  &lb, 0, NULL)) ;
               BeginPath (hdc) ;

               MoveToEx  (hdc, 10 + 30 * i, 25, NULL) ;
               LineTo    (hdc, 20 + 30 * i, 75) ;
               LineTo    (hdc, 30 + 30 * i, 25) ;

               EndPath (hdc) ;
               StrokePath (hdc) ;

               DeleteObject (
                    SelectObject (hdc,
                         GetStockObject (BLACK_PEN))) ;
               MoveToEx  (hdc, 10 + 30 * i, 25, NULL) ;
```

```
                LineTo    (hdc, 20 + 30 * i, 75) ;
                LineTo    (hdc, 30 + 30 * i, 25) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, iMsg, wParam, lParam) ;
}
```

程序使用上述端点和连接点样式画了三条 V 形的宽线段。程序也使用备用黑色画笔画了三条同样的线。这样就将宽线与通常的细线做了比较。结果如图 17-4 所示。

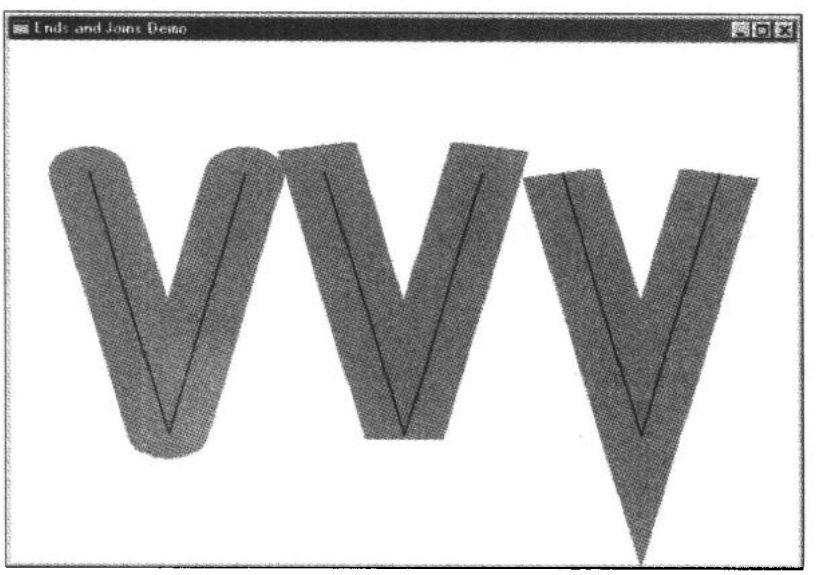


图 17-4　ENDJOIN 显示

现在大家该明白为什么 Windows 支持 StrokePath 函数了：如果分别画两条直线，GDI 不得不在每一条线上使用端点。仅在轨迹定义中，GDI 知道线段是连接的，并使用线段的连接点。

17.6.3　四个示例程序

这究竟有什么好处呢？仔细考虑一下：轮廓字体的字符由一系列坐标值定义，这些坐标定义了直线和转折线。因而，直线及曲线能成为轨迹定义的一部分。

确实可以！程序 17-10 所示的 FONTOUT1 程序对此做了演示。

程序 17-10　FONTOUT1 程序

FONTOUT1.C

```
/*-------------------------
    FONTOUT1.C——Using Path to Outline Font
                (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontOut1") ;
TCHAR szTitle   [] = TEXT ("FontOut1: Using Path to Outline Font") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     static TCHAR  szString [] = TEXT ("Outline") ;
     HFONT         hFont ;
     SIZE          size ;

     hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1440, 0, 0, TRUE) ;

     SelectObject (hdc, hFont) ;

     GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &size) ;

     BeginPath (hdc) ;
     TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
              szString, lstrlen (szString)) ;
     EndPath (hdc) ;

     StrokePath (hdc) ;

     SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
     DeleteObject (hFont) ;
}
```

此程序和本章后面的程序也使用了前面所示的 EZFONT 和 FONTDEMO 文件。

程序创建了 144 磅的 TrueType 字体，并调用 GetTextExtentPoint32 函数获取文本框的大小，然后，调用轨迹定义中的 TextOut 函数使文本在客户区窗口中处于中心的位置。因为对 TextOut 函数的调用是在轨迹支架(即 BeginPath 和 EndPath 调用之间)内进行的，GDI 不立即显示文本。相反，程序将字符轮廓存储在轨迹定义中。

在轨迹支架结束后，FONTOUT1 调用 StrokePath。因为设备描述表中未选入指定的

画笔，所以 GDI 仅仅使用默认画笔绘制字符轮廓，如图 17-5 所示。

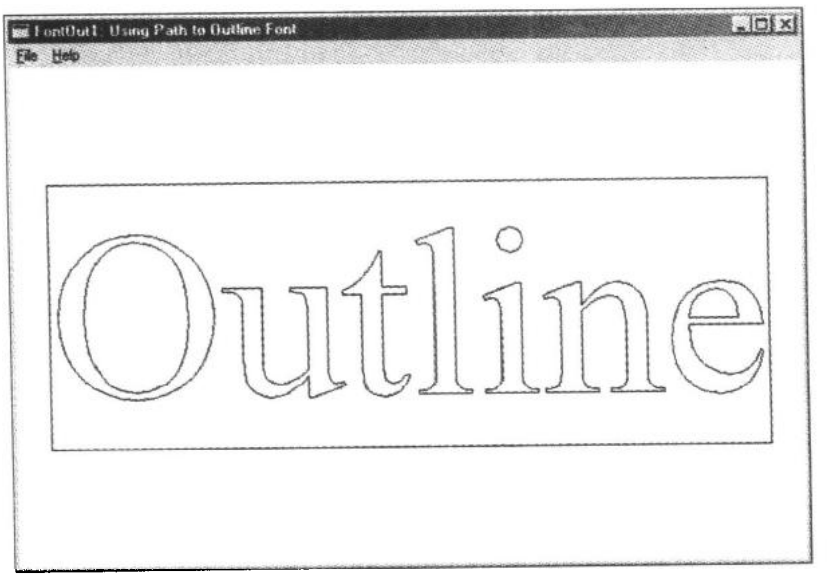


图 17-5　FONTOUT1 显示

现在我们都得到什么呢？我们已经获得了期望的轮廓字符，但是字符串外面为什么会围绕着矩形呢？

回想一下，文本背景模式使用默认的 OPAQUE，而不是 TRANSPARENT。该矩形就是文本框的轮廓。这清晰地演示了在默认的 OPAQUE 模式下 GDI 绘制文本时所使用的两个步骤：首先绘制一个填充的矩形，接着绘制字符。文本框矩形的轮廓也因此成为轨迹的一部分。

使用 ExtCreatePen 函数就能够使用默认画笔以外的东西绘制字体字符的轮廓。程序 17-11 所示的 FONTOUT2 对此做了演示。

程序 17-11　FONTOUT2 程序

FONTOUT2.C

```
/*-----------------------
   FONTOUT2.C——Using Path to Outline Font
                (c) Charles Petzold, 1998
  -----------------------*/

#include <windows.h>
#include "..\\eztest\\ezfont.h"
```

```
TCHAR szAppName [ ] = TEXT ("FontOut2") ;
TCHAR szTitle     [ ] = TEXT ("FontOut2: Using Path to Outline Font") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     static TCHAR  szString [ ] = TEXT ("Outline") ;
     HFONT         hFont ;
     LOGBRUSH      lb ;
     SIZE          size ;

     hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1440, 0, 0, TRUE) ;

     SelectObject (hdc, hFont) ;
     SetBkMode (hdc, TRANSPARENT) ;

     GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &size) ;

     BeginPath (hdc) ;
     TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
                   szString, lstrlen (szString)) ;
     EndPath (hdc) ;

     lb.lbStyle = BS_SOLID ;
     lb.lbColor = RGB (255, 0, 0) ;
     lb.lbHatch = 0 ;

     SelectObject (hdc, ExtCreatePen (PS_GEOMETRIC | PS_DOT,
                              GetDeviceCaps (hdc, LOGPIXELSX) / 24,
                                 &lb, 0, NULL)) ;
     StrokePath (hdc) ;

     DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN))) ;
     SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
     DeleteObject (hFont) ;
}
```

此程序调用 StrokePath 之前创建(并选入设备描述表)一个 3 磅(1/24 英寸)宽的红色点线笔。程序在 Windows NT 下运行时,结果如图 17-6 所示。Windows 98 不支持超过 1 像素宽的非实心笔,因此 Windows 98 将以实心的红色笔绘制。

您也可以使用轨迹定义填充区域。用前面两个程序所示的方法创建轨迹,选择一种填充图案,然后调用 FillPath。可以调用的另一个函数是 StrokeAndFillPath,它绘制轨迹的轮廓,并用一个函数调用对其填充。

StrokeAndFillPath 函数如程序 17-12 所示的 FONTFILL 进行的演示。

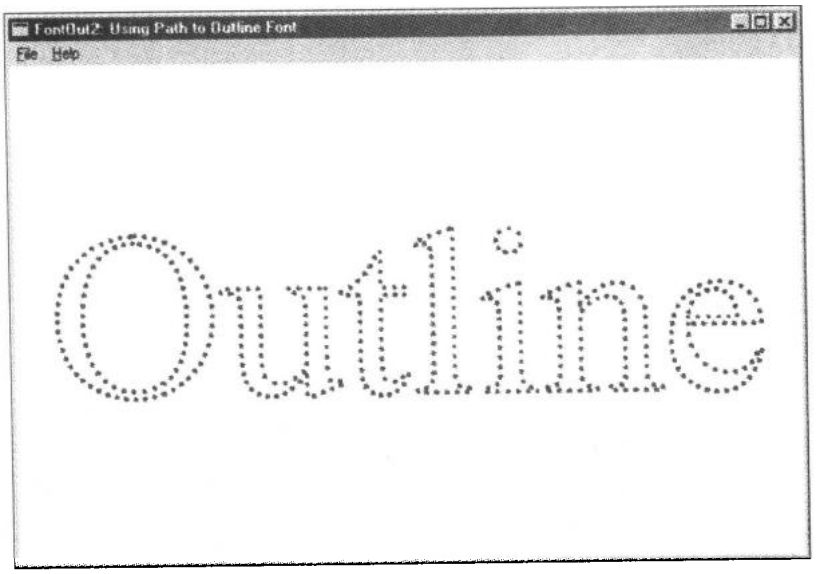


图 17-6　FONTOUT2 显示

程序 17-12　FONTFILL 程序

FONTFILL.C

```
/*------------------------------------
   FONTFILL.C——Using Path to Fill Font
                (c) Charles Petzold, 1998
  ------------------------------------*/

#include <windows.h>
#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontFill") ;
TCHAR szTitle   [] = TEXT ("FontFill: Using Path to Fill Font") ;
void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     static TCHAR  szString [] = TEXT ("Filling") ;
     HFONT         hFont ;
     SIZE          size ;

     hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1440, 0, 0, TRUE) ;
     SelectObject (hdc, hFont) ;
     SetBkMode (hdc, TRANSPARENT) ;

     GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &size) ;

     BeginPath (hdc) ;
     TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
```

```
                szString, lstrlen (szString)) ;
    EndPath (hdc) ;

    SelectObject (hdc, CreateHatchBrush (HS_DIAGCROSS, RGB (255, 0, 0))) ;
    SetBkColor (hdc, RGB (0, 0, 255)) ;
    SetBkMode (hdc, OPAQUE) ;

    StrokeAndFillPath (hdc) ;

    DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRUSH))) ;
    SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
    DeleteObject (hFont) ;
}
```

FONTFILL 使用默认画笔绘制轨迹的轮廓，但使用 HS_DIAGCROSS 样式创建红色的阴影刷子。注意程序在创建轨迹时将背景模式设置为 TRANSPARENT，在填充轨迹时又将其重设为 OPAQUE，这样它可以为区域图案使用蓝色的背景颜色。结果如图 17-7 所示。

图 17-7　FONTFILL 显示

您可能想在本程序中尝试几种变动，观察变动的影响。首先，如果将第一个 SetBkMode 调用变为注释，将得到由图案而不是字符本身覆盖的文本框背景。这通常不是我们实际需要的，但确实可以这样做。

此外，填充字符及将它们用做剪取时，您可能想有效地放弃默认的 ALTERNATE 多边填充模式。我的经验表明：如果使用 WINDING 填充模式，则 TrueType 字体将被构建，从而将不会发生奇怪的现象（例如“O”的内部被填充），但使用 ALTERNATE 模式更安全。

最后，可使用一个轨迹，因此也能用一个 TrueType 字体来定义剪取区域。程序 17-13所示的 FONTCLIP 进行了演示。

程序 17-13　FONTCLIP 程序

FONTCLIP.C

```
/*-----------------------
    FONTCLIP.C——Using Path for Clipping on Font
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>
#include "..\\eztest\\ezfont.h"

TCHAR szAppName [] = TEXT ("FontClip") ;
TCHAR szTitle   [] = TEXT ("FontClip: Using Path for Clipping on Font") ;

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     static TCHAR  szString [] = TEXT ("Clipping") ;
     HFONT         hFont ;
     int           y, iOffset ;
     POINT         pt [4] ;
     SIZE          size ;

     hFont = EzCreateFont (hdc, TEXT ("Times New Roman"), 1200, 0, 0, TRUE) ;
     SelectObject (hdc, hFont) ;

     GetTextExtentPoint32 (hdc, szString, lstrlen (szString), &size) ;

     BeginPath (hdc) ;
     TextOut (hdc, (cxArea - size.cx) / 2, (cyArea - size.cy) / 2,
              szString, lstrlen (szString)) ;
     EndPath (hdc) ;

          // Set clipping area

     SelectClipPath (hdc, RGN_COPY) ;

          // Draw Bezier splines

     iOffset = (cxArea + cyArea) / 4 ;

     for (y = -iOffset ; y < cyArea + iOffset ; y++)
     {
          pt[0].x = 0 ;
          pt[0].y = y ;
```

```
            pt[1].x = cxArea / 3 ;
            pt[1].y = y + iOffset ;

            pt[2].x = 2 * cxArea / 3 ;
            pt[2].y = y - iOffset ;

            pt[3].x = cxArea ;
            pt[3].y = y ;

            SelectObject (hdc, CreatePen (PS_SOLID, 1,
                RGB (rand () % 256, rand () % 256, rand () % 256))) ;

            PolyBezier (hdc, pt, 4) ;

            DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN))) ;
        }

        DeleteObject (SelectObject (hdc, GetStockObject (WHITE_BRUSH))) ;
        SelectObject (hdc, GetStockObject (SYSTEM_FONT)) ;
        DeleteObject (hFont) ;
    }
```

程序中有意不使用 SetBkMode 调用，以实现不同的效果。程序在轨迹支架中绘制一些文本，然后调用 SelectClipPath，接着使用随机颜色绘制一系列贝塞尔样条曲线。

如果 FONTCLIP 程序使用 TRANSPARENT 选项调用 SetBkMode，贝塞尔曲线将被限制在字符轮廓的内部。在默认的 OPAQUE 选项的背景模式下，剪取区域被限制在文本框内部而不是文本内部。如图 17-8 所示。

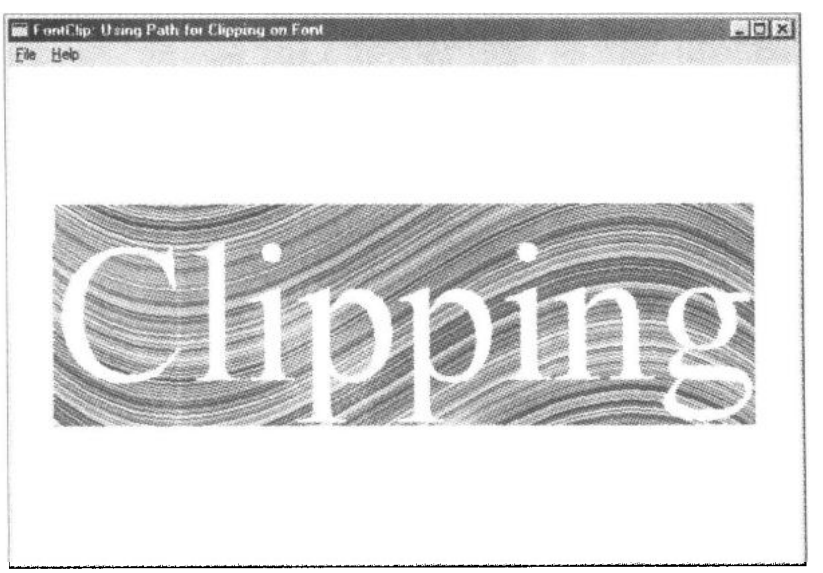


图 17-8　FONTCLIP 显示

您或许会想在 FONTCLIP 中插入 SetBkMode 调用来观察 TRANSPARENT 选项的变化。

FONTDEMO 外壳程序允许您打印并显示这些效果，甚至允许您尝试自己的一些特殊效果。

第十八章

元文件

元文件和矢量图形的关系，就像位图和光栅图形的关系一样。位图通常是来自现实世界的图像，而元文件是人为创建的，并且通常是在计算机程序的协助下创建的。元文件由一系列与图形函数调用等价的二进制记录组成，这些记录一般用于绘制直线、曲线、填充的区域和文本等。

“画图(Paint)”程序创建位图，而“绘图(draw)”程序创建元文件。在优秀的绘图程序中，您可以很容易地“抓住”某个独立的图形对象(例如一条直线)，并将它移动到任意不同的位置。这是因为组成图形的每个成员都是以单独的记录存储的。在画图程序中，这种功能是不可能实现的——通常您都会被局限于删除或插入位图矩形块的操作中。

由于元文件以图形绘制命令描述图像，因此可以对图像进行缩放而不会失真。位图则不然，如果以两倍大小来显示位图，您却无法得到两倍的分辨率，而只是在水平和垂直方向上重复位图的位。

元文件可以转换为位图，但是会丢失一些信息：组成元文件的图形对象将不再是独立的了，而是被合并进大的图像。将位图转换为元文件要艰难得多，一般仅限于非常简单的图像，而且它需要大量处理来分析边界和轮廓。而元文件可以包含绘制位图的命令。

虽然元文件可以作为剪贴画保存在磁盘上，但是它们大多用于程序通过剪贴板共享图片的情况。由于元文件将图片描述为图像函数调用的集合，因而它们既比位图占用更少的空间，又比位图更与设备无关。

Microsoft Windows 支持两种元文件格式和相应的两组函数。我首先讨论从 Windows 1.0 到当前的 32 位 Windows 版本都支持的元文件函数，然后讨论为 32 位 Windows 系统开发的“增强型元文件”。增强型元文件在原有元文件的基础上有了一些提高，应该尽可能地加以利用。

18.1 老式元文件格式

元文件既能够临时存储在内存中，也能够以文件的形式存储在磁盘上。对应用程序来说，两者区别不大，尤其是由 Windows 来处理从基于磁盘的元文件中保存和加载数据的文件 I/O 时，更是如此。

18.1.1 内存元文件的简单利用

如果通过调用 CreateMetaFile 函数来创建元文件设备描述表，就以早期的格式创建一个元文件，然后就可以使用大部分 GDI 绘图函数在该元文件设备描述表上进行绘图。这些 GDI 调用并不在任何具体的设备上绘图，相反，它们被存储在元文件中。当关闭元文件设备描述表时，会得到元文件的句柄。这时就可以在某个具体的设备描述表上“播放”这个元文件，这与直接运行元文件中 GDI 函数的效果等同。

CreateMetaFile 只有一个参数，它可以是 NULL 或文件名。如果是 NULL，则元文件存储在内存中；如果是文件名（扩展名 .WMF 代表“Windows Metafile”），则元文件存储在磁盘文件中。

程序 18-1 中的 METAFILE 显示了在 WM_CREATE 消息期间创建内存元文件的方法，并在 WM_PAINT 消息期间将图像显示 100 遍。

程序 18-1　METAFILE 程序

METAFILE.C

```
/*-----------------------
   METAFILE.C——Metafile Demonstration Program
               (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName [] = TEXT ("Metafile") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
```

```
    wndclass.lpfnWndProc   = WndProc ;
    wndclass.cbClsExtra    = 0 ;
    wndclass.cbWndExtra    = 0 ;
    wndclass.hInstance     = hInstance ;
    wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName  = NULL ;
    wndclass.lpszClassName = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Metafile Demonstration"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HMETAFILE  hmf ;
    static int        cxClient, cyClient ;
    HBRUSH            hBrush ;
    HDC               hdc, hdcMeta ;
    int               x, y ;
    PAINTSTRUCT       ps ;

    switch (message)
    {
    case WM_CREATE:
        hdcMeta = CreateMetaFile (NULL) ;
```

```
            hBrush   = CreateSolidBrush (RGB (0, 0, 255)) ;
            Rectangle (hdcMeta, 0, 0, 100, 100) ;

            MoveToEx (hdcMeta,   0,   0, NULL) ;
            LineTo   (hdcMeta, 100, 100) ;
            MoveToEx (hdcMeta,   0, 100, NULL) ;
            LineTo   (hdcMeta, 100,   0) ;

            SelectObject (hdcMeta, hBrush) ;
            Ellipse (hdcMeta, 20, 20, 80, 80) ;

            hmf = CloseMetaFile (hdcMeta) ;

            DeleteObject (hBrush) ;
            return 0 ;

        case WM_SIZE:
            cxClient = LOWORD (lParam) ;
            cyClient = HIWORD (lParam) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            SetMapMode (hdc, MM_ANISOTROPIC) ;
            SetWindowExtEx (hdc, 1000, 1000, NULL) ;
            SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;

            for (x = 0 ; x < 10 ; x++)
            for (y = 0 ; y < 10 ; y++)
            {
                SetWindowOrgEx (hdc, -100 * x, -100 * y, NULL) ;
                PlayMetaFile (hdc, hmf) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            DeleteMetaFile (hmf) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

这个程序演示了在使用内存元文件时所涉及的4个元文件函数的用法。第一个是CreateMetaFile，在处理 WM_CREATE 消息期间用 NULL 参数调用该函数，并返回元文件

设备描述表的句柄，然后，METAFILE 利用这个元文件 DC 来绘制两条直线和一个蓝色椭圆。这些函数调用以二进制形式存储在元文件中。CloseMetaFile 函数返回元文件的句柄。因为以后还要用到该元文件句柄，所以把它存储在静态变量中。

该元文件包含 GDI 函数调用的二进制表示，它们是两个 MoveToEx 调用、两个 LineTo 调用、一个 SelectObject 调用（指定蓝色画刷），以及一个 Ellipse 调用。坐标没有指定任何映射方式或转换，它们只是作为数字被存储在元文件中。

在 WM_PAINT 消息期间，METAFILE 设置一种映射方式，并调用 PlayMetaFile 在窗口中绘制对象 100 次。元文件中函数调用的坐标按照目标设备描述表的当前变换方式加以解释。在调用 PlayMetaFile 时，事实上是在重复地调用最初在 WM_CREATE 消息期间创建元文件时在 CreateMetaFile 和 CloseMetaFile 之间所做的所有调用。

和任何 GDI 对象一样，元文件对象也应该在程序终止前被删除。这是在 WM_DESTROY 消息期间用 DeleteMetaFile 函数实现的。

METAFILE 程序的结果如图 18-1 所示。

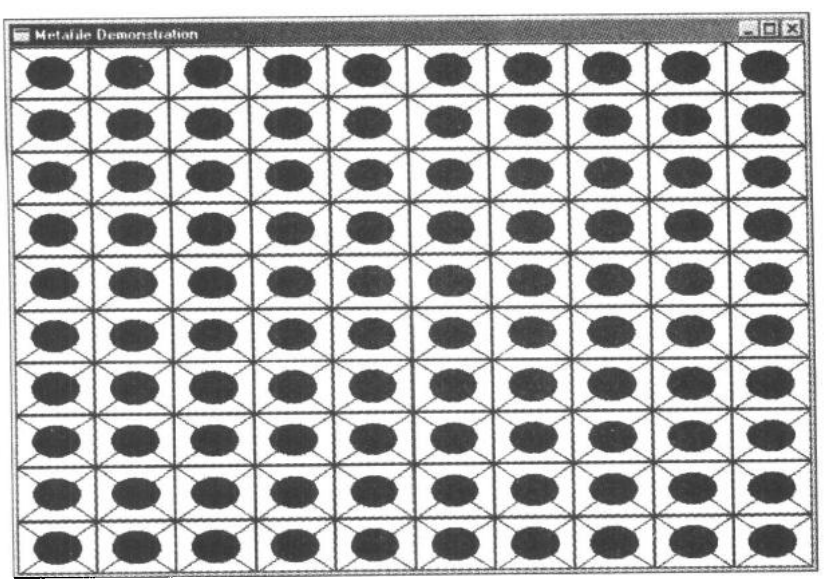


图 18-1　METAFILE 程序运行结果显示

18.1.2　将元文件保存在磁盘上

在上面的例子中，CreateMetaFile 的 NULL 参数表示要创建存储在内存中的元文件。我们也可以创建作为文件存储在磁盘上的元文件，这种方法对于大的元文件比较合适，因为这样可以节省内存空间。而另一方面，每次使用磁盘上的元文件时，都需要访问磁盘。

要把 METAFILE 转换为使用基于磁盘的元文件，需要把 CreateMetaFile 的 NULL 参数替换为文件名。在 WM_CREATE 处理结束时，可以用元文件句柄来调用 DeleteMetaFile，这样句柄被删除，但是磁盘文件仍然被保存着。

在处理 WM_PAINT 消息期间，可以通过调用 GetMetaFile 来获取磁盘文件的元文件句柄：

```
hmf = GetMetaFile (szFileName) ;
```

现在就可以像前面那样显示这个元文件。在 WM_PAINT 消息处理结束时，可以用下面的语句删除元文件句柄：

```
DeleteMetaFile (hmf) ;
```

在开始处理 WM_DESTROY 消息时，不必删除元文件，因为它已经在 WM_CREATE 消息和每个 WM_PAINT 消息结束时被删除了，但是仍然需要删除磁盘文件：

```
DeleteFile (szFileName) ;
```

当然，除非您想保存该文件。

正如在第十章讨论过的，元文件也可以作为用户自定义资源。您可以简单地把它当作数据块加载。如果有一块包含元文件内容的数据，那么您可以使用

```
hmf = SetMetaFileBitsEx (iSize, pData) ;
```

来创建元文件。SetMetaFileBitsEx 有一个对应的函数——GetMetaFileBitsEx，此函数将元文件的内容复制到内存块中。

18.1.3　老式元文件与剪贴板

老式元文件有个讨厌的缺陷。如果具有老式元文件的句柄，那么在显示元文件时，如何确定它的大小呢？除非您深入元文件的内部，否则无法得知。

此外，当程序从剪贴板获取老式元文件时，如果元文件被定义为在 MM_ISOTROPIC 或 MM_ANISOTROPIC 映射方式下显示，则此程序在使用该元文件时具有最大程度的灵活性。程序接受元文件后，就可以在显示它之前简单地通过设置视口的范围来缩放图像。然而，如果元文件内的映射方式被设置为 MM_ISOTROPIC 或 MM_ANISOTROPIC，则接受元文件的程序将无法继续运行。程序仅能在显示元文件之前或之后进行 GDI 调用，不允许在显示元文件当中进行 GDI 调用。

为了解决这些问题，老式元文件句柄不直接放入剪贴板供其他程序获取，而是作为“元文件图片”（METAFILEPICT 结构类型）的一部分。此结构使得从剪贴板上获取元文件图片的程序能够在显示元文件之前设置映射方式和视口范围。

METAFILEPICT 结构的长度为 16 个字节，定义如下：

```
typedef struct tagMETAFILEPICT
{
    LONG mm ;              // mapping mode
    LONG xExt ;            // width of the metafile image
    LONG yExt ;            // height of the metafile image
    LONG hMF ;             // handle to the metafile
}
METAFILEPICT ;
```

对于除 MM_ISOTROPIC 和 MM_ANISOTROPIC 以外的所有映射方式，图像大小用 xExt 和 yExt 值表示，其单位是由 mm 给出的映射方式的单位。利用这种信息，从剪贴板复制元文件图片结构的程序就能够确定在显示元文件时所需的显示空间。创建元文件的程序可以将这些值设置为输入元文件的 GDI 绘制函数中使用的最大的 x 坐标和 y 坐标值。

在 MM_ISOTROPIC 和 MM_ANISOTROPIC 映射方式下，xExt 和 yExt 字段有不同的功能。我们在第五章中曾介绍过一个程序，该程序为了在 GDI 函数中使用与图像实际尺寸无关的逻辑单位而采用 MM_ISOTROPIC 或 MM_ANISOTROPIC 映射方式。当程序只想保持纵横比而忽略视图表面的大小时，采用 MM_ISOTROPIC 模式；反之，当不需要考虑纵横比时，采用 MM_ANISOTROPIC 模式。您也许还记得，第五章中在程序将映射方式设置为 MM_ISOTROPIC 或 MM_ANISOTROPIC 后，通常调用 SetWindowExtEx 和 SetViewportExtEx。SetWindowExtEx 调用使用逻辑单位来指定程序在绘制时使用的单位，而 SetViewportExtEx 调用则使用基于视图表面的大小（例如窗口客户区的大小）的设备单位。

如果程序为剪贴板创建了 MM_ISOTROPIC 或 MM_ANISOTROPIC 方式的元文件，则该元文件本身不应包含对 SetViewportExtEx 的调用，因为该调用中的设备单位应该基于创建元文件的程序的显示表面，而不是基于从剪贴板读取并显示元文件的程序的显示表面。从剪贴板获取元文件的程序可以利用 xExt 和 yExt 值来设置合适的视口范围以便显示元文件。但是，当映射方式是 MM_ISOTROPIC 或 MM_ANISOTROPIC 时，元文件本身包含设置窗口范围的调用。元文件内的 GDI 绘图函数的坐标基于这些窗口的范围。

创建元文件和元文件图片遵循以下规则：

- 设置 METAFILEPICT 结构的 mm 字段来指定映射方式。
- 对于 MM_ISOTROPIC 和 MM_ANISOTROPIC 以外的映射方式，xExt 与 yExt 字段设置为图像的宽和高，单位与 mm 字段相对应。对于在 MM_ISOTROPIC 或 MM_ANISOTROPIC 方式下显示的元文件，工作要复杂一些。在 MM_ANISOTROPIC 模式下，当

程序既不依据一种大小也不依据纵横比时，xExt 和 yExt 的值均为 0。在这两种模式下，如果 xExt 和 yExt 的值为正数，它们就指出了以 0.01 mm 为单位（MM_HIMETRIC 单位）表示的图像的宽度和高度。在 MM_ISOTROPIC 方式下，如果 xExt 和 yExt 为负值，它们就指出了图像的纵横比，而不是其大小。

- 在 MM_ISOTROPIC 和 MM_ANISOTROPIC 映射方式下，元文件本身含有对 SetWindowExtEx 的调用，也可能有对 SetWindowOrgEx 的调用，即创建元文件的程序在元文件设备描述表中调用这些函数。元文件一般不会包含对 SetMapMode、SetViewportExtEx 或 SetViewportOrgEx 的调用。
- 元文件应该是基于内存的元文件，而不是基于磁盘的元文件。

这里有一段示例代码，它创建元文件并将其复制到剪贴板中。如果元文件使用 MM_ISOTROPIC 或 MM_ANISOTROPIC 映射方式，则该元文件的第一个调用应该设置窗口范围。（在其他模式中窗口的大小是固定的。）无论在哪种模式下，窗口的位置应如下设置：

```
hdcMeta = CreateMetaFile (NULL) ;
SetWindowExtEx (hdcMeta, ...) ;
SetWindowOrgEx (hdcMeta, ...) ;
```

元文件绘图函数中的坐标基于这些窗口范围和窗口原点。当程序使用 GDI 调用在元文件设备描述表中绘制完成后，关闭元文件得到元文件句柄：

```
hmf = CloseMetaFile (hdcMeta) ;
```

该程序还需要定义指向 METAFILEPICT 类型结构的指针，并为此结构分配一块全局内存：

```
GLOBALHANDLE hGlobal ;
LPMETAFILEPICT pMFP ;
[ Other program lines ]
hGlobal = GlobalAlloc (GHND | GMEM_SHARE, sizeof (METAFILEPICT)) ;
pMFP = (LPMETAFILEPICT) GlobalLock (hGlobal) ;
```

接着，程序设置该结构的 4 个字段：

```
pMFP -> mm    = MM_... ;
pMFP -> xExt  = ... ;
pMFP -> yExt  = ... ;
pMFP -> hMF   = hmf ;

GlobalUnlock (hGlobal) ;
```

然后，程序将包含有元文件图片的全局内存块传送给剪贴板：

```
OpenClipboard (hwnd) ;
EmptyClipboard () ;
SetClipboardData (CF_METAFILEPICT, hGlobal) ;
CloseClipboard () ;
```

完成这些调用后，hGlobal 句柄(包含元文件图片结构的内存块)和 hmf 句柄(元文件本身)对于创建它们的程序失效。

现在来看困难的部分。当程序从剪贴板获取元文件并显它时必须完成下列步骤：

1. 程序利用元文件图片结构的 mm 字段设置映射方式。
2. 对于 MM_ISOTROPIC 或 MM_ANISOTROPIC 以外的映射方式，程序用 xExt 和 yExt 值设置剪贴矩形或简单地设置图像大小。对于 MM_ISOTROPIC 和 MM_ANISOTROPIC 映射方式，程序使用 xExt 和 yExt 来设置视口范围。
3. 然后，程序运行元文件。

下面是代码，首先打开剪贴板，得到元文件图片结构句柄并将其锁定：

```
OpenClipboard (hwnd) ;
hGlobal = GetClipboardData (CF_METAFILEPICT) ;
pMFP = (LPMETAFILEPICT) GlobalLock (hGlobal) ;
```

现在可以保存当前设备描述表的属性，并将映射方式设置为结构中的 mm 值：

```
SaveDC (hdc) ;
SetMappingMode (pMFP -> mm) ;
```

如果映射方式不是 MM_ISOTROPIC 或 MM_ANISOTROPIC，则可以用 xExt 和 yExt 的值设置剪贴矩形。由于这两个值是逻辑单位，必须用 LPtoDP 将其转换为用于剪贴矩形的设备单位的坐标，也可以简单地保存这些值以掌握图像的大小。

对于 MM_ISOTROPIC 或 MM_ANISOTROPIC 映射方式，xExt 和 yExt 用来设置视口范围。下面有一个用来完成此项任务的函数，如果 xExt 和 yExt 没有建议的大小，则该函数假定 cxClient 和 cyClien 分别表示元文件显示区域的像素高度和宽度。

```
void PrepareMetaFile (HDC hdc, LPMETAFILEPICT pmfp,
                      int cxClient, int cyClient)
{
     int xScale, yScale, iScale ;

     SetMapMode (hdc, pmfp -> mm) ;

     if (pmfp -> mm == MM_ISOTROPIC || pmfp -> mm == MM_ANISOTROPIC)
     {
          if (pmfp -> xExt == 0)
               SetViewportExtEx (hdc, cxClient, cyClient, NULL) ;
          else if (pmfp -> xExt > 0)
```

```
                SetViewportExtEx (hdc,
                    pmfp -> xExt * GetDeviceCaps (hdc, HORZRES) /
                                   GetDeviceCaps (hdc, HORZSIZE) / 100),
                    pmfp -> yExt * GetDeviceCaps (hdc, VERTRES) /
                                   GetDeviceCaps (hdc, VERTSIZE) / 100),
                    NULL) ;

            else if (pmfp -> xExt < 0)
            {
                xScale = 100 * cxClient * GetDeviceCaps (hdc, HORZSIZE) /
                               GetDeviceCaps (hdc, HORZRES) / - pmfp -> xExt ;
                yScale = 100 * cyClient * GetDeviceCaps (hdc, VERTSIZE) /
                               GetDeviceCaps (hdc, VERTRES) / - pmfp -> yExt ;
                iScale = min (xScale, yScale) ;

                SetViewportExtEx (hdc,
                    - pmfp -> xExt * iScale * GetDeviceCaps (hdc, HORZRES) /
                                     GetDeviceCaps (hdc, HORZSIZE) / 100,
                    - pmfp -> yExt * iScale * GetDeviceCaps (hdc, VERTRES) /
                                     GetDeviceCaps (hdc, VERTSIZE) / 100,
                    NULL) ;
            }
        }
    }
```

上面的代码假设 xExt 和 yExt 都为 0、大于 0 或小于 0。如果范围为 0，范围表示没有建议的大小或纵横比，视口范围设置为显示元文件的区域。如果范围大于 0，那么 xExt 和 yExt 的值代表图像的建议大小，单位是 0.01 mm。GetDeviceCaps 函数用来确定每 0.01 mm 中包含的像素数，并且该值与元文件图片结构的范围值相乘。如果范围小于 0，那么 xExt 和 yExt 的值表示建议的纵横比而不是建议的大小。iScale 的值首先根据对应 cxClient 和 cyClient 的 mm 表示的纵横比计算出来，该缩放因子用于设置像素单位的视口范围。

完成了上述工作后，可以设置视口原点，显示元文件，并恢复设备描述表：

```
PlayMetaFile (pMFP -> hMF) ;
RestoreDC (hdc, -1) ;
```

然后，对内存块解锁并关闭剪贴板：

```
GlobalUnlock (hGlobal) ;
CloseClipboard () ;
```

如果程序使用增强型元文件就可以省去这项工作。当某个应用程序将这些格式放入剪贴板而另一个程序却要求从剪贴板中获得其他格式时，Windows 剪贴板会自动在老式元文件和增强型元文件之间进行格式转换。

18.2 增强型元文件

“增强型元文件”格式是在32位Windows版本中引入的。它包含一组新的函数调用、一些新的数据结构、一种新的剪贴板格式，以及一个新的文件扩展名.EMF。

这种新的元文件格式最重要的改进是加入可通过函数调用获取的更丰富的头信息，这种头信息用来帮助应用程序显示元文件图像。

有些增强型元文件函数使您能够在增强型元文件(EMF)格式和老式元文件格式(也称作Windows元文件，WMF格式)之间来回转换。当然，这种转换很可能遇到麻烦，因为老式元文件格式并不支持某些新的32位图形特性，例如路径。

18.2.1 基本过程

程序18-2所示的EMF1创建并显示增强型元文件。

程序18-2　EMF1程序

EMF1.C

```
/* ------------------
    EMF1.C——Enhanced Metafile Demo #1
            (c) Charles Petzold, 1998
   ------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpszCmdLine, int nCmdShow)
{
    static TCHAR  szAppName[] = TEXT ("EMF1") ;
    HWND          hwnd ;
    MSG           msg ;
    WNDCLASS      wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
```

```
    wndclass.lpszMenuName  = NULL ;
    wndclass.lpszClassName = szAppName ;

    if (!RegisterClass (&wndclass))
    {
         MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                     szAppName, MB_ICONERROR) ;
         return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #1"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, nCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HENHMETAFILE  hemf ;
    HDC                  hdc, hdcEMF ;
    PAINTSTRUCT          ps ;
    RECT                 rect ;

    switch (message)
    {
    case WM_CREATE:
        hdcEMF = CreateEnhMetaFile (NULL, NULL, NULL, NULL) ;

        Rectangle  (hdcEMF, 100, 100, 200, 200) ;

        MoveToEx (hdcEMF, 100, 100, NULL) ;
        LineTo    (hdcEMF, 200, 200) ;

        MoveToEx (hdcEMF, 200, 100, NULL) ;
        LineTo    (hdcEMF, 100, 200) ;

        hemf = CloseEnhMetaFile (hdcEMF) ;
```

```
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;

            rect.left   =     rect.right / 4 ;
            rect.right  = 3 * rect.right / 4 ;
            rect.top    =     rect.bottom / 4 ;
            rect.bottom = 3 * rect.bottom / 4 ;

            PlayEnhMetaFile (hdc, hemf, &rect) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            DeleteEnhMetaFile (hemf) ;

            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在 EMF1 的窗口过程处理 WM_CREATE 消息期间，程序首先通过调用 CreateEnhMetaFile 来创建增强型元文件。该函数有 4 个参数，但可以把它们都设为 NULL。稍后我将说明这 4 个参数在非 NULL 情况下的使用方法。

和 CreateMetaFile 一样，CreateEnhMetaFile 函数返回特定的设备描述表句柄。该程序利用这个句柄绘制一个矩形和该矩形的两条对角线。这些函数调用及其参数被转换为二进制的形式存储在元文件中。

最后通过对 CloseEnhMetaFile 函数的调用结束了增强型元文件的创建，并返回指向它的句柄。该文件句柄存储在 HENHMETAFILE 类型的静态变量中。

在 WM_PAINT 消息期间，EMF1 以 RECT 结构获取程序的客户区窗口大小。通过调整结构中的 4 个字段，使该矩形的长和宽为客户区窗口长和宽的一半，并位于窗口的中央，然后 EMF1 调用 PlayEnhMetaFile，该函数的第一个参数是窗口的设备描述表句柄，第二个参数是该增强型元文件的句柄，第三个参数是指向 RECT 结构的指针。

在元文件的创建过程中，GDI 得出整个元文件图像的尺寸。在本例中，图像的长和宽均为 100 个单位。在元文件的显示过程中，GDI 将图像拉伸，以适应 PlayEnhMetaFile 函数指定的矩形大小。EMF1 在 Windows 下运行的三个实例如图 18-2 所示。

最后，在 WM_DESTROY 消息期间，EMF1 调用 DeleteEnhMetaFile 删除元文件。

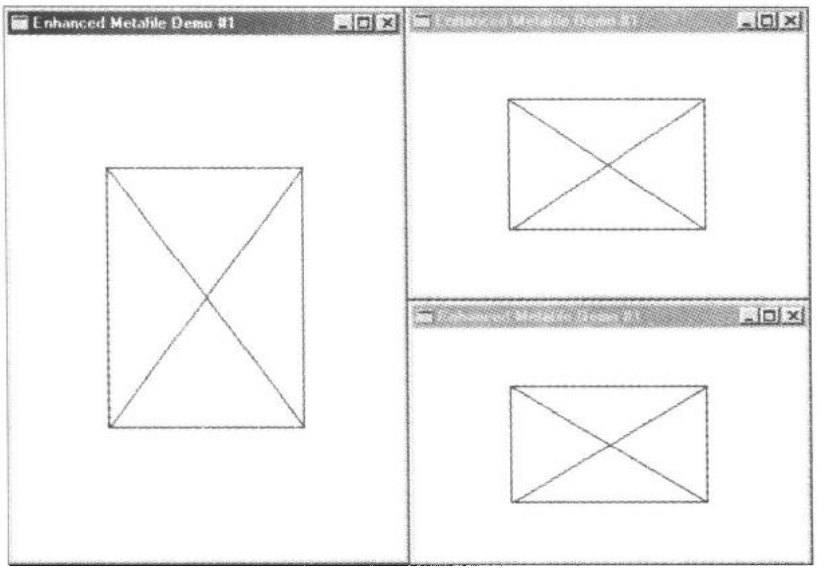


图 18-2　EMF1 显示

让我们总结一下从 EMF1 程序学到的一些东西。

首先，该程序在创建增强型元文件时，画矩形和直线的函数使用的坐标并不是实际意义上的坐标。您可以将它们同时翻倍或都减去某个常数，而其结果不会改变。这些坐标只是在定义图像时说明彼此间的对应关系。

其次，为适应传递给 PlayEnhMetaFile 函数的矩形，图像会被拉伸。因此图像可能会变形，如图18-2所示。尽管元文件坐标指出该图像是正方形的，但一般情况下我们却得不到这样的图像。而在某些时候，这又正是我们想要得到的图像。例如，将图像嵌入一段字处理文本中时，可能会要求用户为图像指定矩形，并且确保整个图像恰好位于矩形中而不浪费空间。这样，用户通过适当调整矩形的大小来得到正确的纵横比。

然而有时候，您也许希望保留图像最初的纵横比，因为这一点对于表现视觉信息尤为重要。例如，警察的嫌疑犯草图既不能比原型胖也不能比原型瘦。或者您希望保留原来图像的度量尺寸，图像必须是两英寸高，否则就不能正常显示。在这种情况下，保留图像的原来尺寸就非常重要了。

同时也要注意元文件中画出的那些对角线似乎没有与矩形顶点相交。这是由于 Windows 在元文件中存储矩形坐标的方式造成的。稍后，会阐述解决这个问题的方法。

18.2.2　揭开内幕

如果看一看元文件的内容会对元文件工作的方式有一个更好的理解。如果有一个基于磁盘的元文件，这将很容易做到，程序 18-3 中的 EMF2 程序创建了一个元文件。

程序 18-3　EMF2 程序

EMF2.C

```
/*-------------------
   EMF2.C——Enhanced Metafile Demo #2
            (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    LPSTR lpszCmdLine, int nCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("EMF2") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #2"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, nCmdShow) ;
```

```
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    HDC             hdc, hdcEMF ;
    HENHMETAFILE    hemf ;
    PAINTSTRUCT     ps ;
    RECT            rect ;

    switch (message)
    {
    case WM_CREATE:
        hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf2.emf"), NULL,
                                    TEXT ("EMF2 \ 0EMF Demo #2 \ 0")) ;
        if (!hdcEMF)
            return 0 ;

        Rectangle  (hdcEMF, 100, 100, 200, 200) ;

        MoveToEx  (hdcEMF, 100, 100, NULL) ;
        LineTo    (hdcEMF, 200, 200) ;

        MoveToEx  (hdcEMF, 200, 100, NULL) ;
        LineTo    (hdcEMF, 100, 200) ;

        hemf = CloseEnhMetaFile (hdcEMF) ;

        DeleteEnhMetaFile (hemf) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;

        rect.left    =       rect.right / 4 ;
        rect.right   = 3 *   rect.right / 4 ;
        rect.top     =       rect.bottom / 4 ;
        rect.bottom  = 3 *   rect.bottom / 4 ;
```

```
            if (hemf = GetEnhMetaFile (TEXT ("emf2.emf")))
            {
                PlayEnhMetaFile (hdc, hemf, &rect) ;
                DeleteEnhMetaFile (hemf) ;
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在 EMF1 程序中，CreateEnhMetaFile 函数的所有参数均被设置为 NULL。在 EMF2 中，第一个参数仍旧设置为 NULL，该参数还可以是设备描述表句柄。GDI 使用该参数在元文件头中插入度量信息，很快我会讨论它。如果该参数为 NULL，则 GDI 认为度量信息是由视频设备描述表确定的。

CreateEnhMetaFile 函数的第二个参数是文件名。如果该参数为 NULL(在 EMF1 中为 NULL，但在 EMF2 中不为 NULL)，则该函数创建内存元文件。EMF2 创建名为 EMF2.EMF 的基于磁盘的元文件。

函数的第三个参数是 RECT 结构的地址，它指出了以 0.01 mm 为单位的元文件总的大小。这是元文件头中极其重要的信息(这是早期的 Windows 元文件格式的缺陷之一)。如果该参数为 NULL，GDI 会计算出尺寸。我比较喜欢让操作系统替我做这些事，所以将该参数设置为 NULL。当应用程序对性能要求比较严时，需要使用该参数以避免 GDI 执行额外的工作。

最后的参数是描述该元文件的字符串。该字符串分为两部分：第一部分是应用软件的名称(不一定是程序的文件名)，以 NULL 结尾；第二部分描述图像内容的文字，以两个 NULL 结尾。例如用 C 语言中的符号“\0”作为 NULL 字符，则该描述串可以是“LoonyCad V6.4\0Flying Frogs\0\0”。由于在 C 语言中通常在引用的字串末尾放入一个 NULL 字符，所以如 EMF2 所示，在末尾仅需一个“\0”。

创建完元文件后，与 EMF1 一样，EMF2 也通过利用由 CreateEnhMetaFile 函数返回的设备描述表句柄进行一些 GDI 函数调用，然后程序调用 CloseEnhMetaFile 删除设备描述表句柄，并获取完成的元文件的句柄。

然后，仍在 WM_CREATE 消息处理期间，EMF2 做了一些 EMF1 没有做的事情：在获得元文件句柄之后，程序调用 DeleteEnhMetaFile。该操作释放用于保存元文件的所有内存资源。然而，基于磁盘的元文件仍然保留在磁盘中(如果愿意，您可以使用如 DeleteFile 的文件删除函数来删除文件)。注意元文件句柄并不像 EMF1 中那样存储在静态变

量中，这意味着在消息之间不需要保存它。

现在，为了使用该元文件，EMF2 需要访问磁盘文件。这是在 WM_PAINT 消息期间通过调用 GetEnhMetaFile 实现的。元文件的文件名是该函数的唯一参数，该函数返回元文件句柄。和 EMF1 一样，EMF2 将这个文件句柄传递给 PlayEnhMetaFile 函数。该元文件图像在 PlayEnhMetaFile 函数的最后一个参数所指定的矩形中显示。与 EMF1 不同的是，EMF2 在 WM_PAINT 消息结束之前就删除该元文件。此后每当位于 WM_PAINT 消息期间，EMF2 都再次读取元文件，显示并删除它。

要记住，对元文件的删除操作仅是释放了用以保存元文件的内存资源而已，磁盘元文件甚至在程序执行结束后还保留在磁盘上。

由于 EMF2 留下了基于磁盘的元文件，您可以看一看它的内容。图 18-3 显示了该程序创建的 EMF2.EMF 文件的一堆十六进制代码。

图 18-3 所示的元文件是 EMF2 在 Microsoft Windows NT 4 下，视频显示器的分辨率为 1024×768 的条件下创建的。同一程序在 Windows 98 下创建的元文件会比前者少 12 个字节，这一点我一会儿讨论。同样，视频显示器的分辨率也将影响元文件头的某些信息。

```
0000  01 00 00 00 88 00 00 00 64 00 00 00 64 00 00 00  ........d...d...
0010  C8 00 00 00 C8 00 00 00 35 0C 00 00 35 0C 00 00  ....`.....5...5...
0020  6A 18 00 00 6A 18 00 00 20 45 4D 46 00 00 01 00  j...j... EMF....
0030  F4 00 00 00 07 00 00 00 01 00 00 00 12 00 00 00  ................
0040  64 00 00 00 00 00 00 00 00 00 04 00 00 00 03 00  d...............
0050  40 01 00 00 F0 00 00 00 00 00 00 00 00 00 00 00  @...............
0060  00 00 00 00 45 00 4D 00 46 00 32 00 00 00 45 00  ....E.M.F.2...E.
0070  4D 00 46 00 20 00 44 00 65 00 6D 00 6F 00 20 00  M.F. .D.e.m.o. .
0080  23 00 32 00 00 00 00 00 2B 00 00 00 18 00 00 00  #.2.....+.......
0090  63 00 00 00 63 00 00 00 C6 00 00 00 C6 00 00 00  c...c...........
00A0  1B 00 00 00 10 00 00 00 64 00 00 00 64 00 00 00  ........d...d...
00B0  36 00 00 00 10 00 00 00 C8 00 00 00 C8 00 00 00  6...............
00C0  1B 00 00 00 10 00 00 00 C8 00 00 00 64 00 00 00  ............d...
00D0  36 00 00 00 10 00 00 00 64 00 00 00 C8 00 00 00  6.......d.......
00E0  0E 00 00 00 14 00 00 00 00 00 00 00 10 00 00 00  ................
00F0  14 00 00 00                                      ....
```

图 18-3　EMF2.EMF 的十六进制代码

增强型元文件格式使我们对元文件的工作方式有更深刻的理解。增强型元文件由可变长度的记录组成，这些记录的通常格式由 ENHMETARECORD 结构说明，它在 WINGDI.H 头文件中定义如下：

```
typedef struct tagENHMETARECORD
{
    DWORD iType ;              // record type
    DWORD nSize ;              // record size
    DWORD dParm [1] ;          // parameters
}
ENHMETARECORD ;
```

当然，那个有一个元素的数组指出了数组元素的变量数。参数的数量取决于记录类型。iType 字段可以是定义在 WINGDI.H 文件中以前缀 EMR_开始的近百个常量之一。nSize 字段是总记录的大小，包括 iType 和 nSize 字段，以及一个或多个 dParm 字段。

带着这些知识，我们看一下图 18-3。第一个字段类型为 0x00000001，大小为 0x00000088，所以它占据文件的前 136 个字节。记录类型为 1 表示常量 EMR_HEADER。我们不妨把对头记录的讨论往后搁，先跳到位于第一个记录末尾的偏移量 0x0088 处。

后面的 5 个记录与 EMF2 创建元文件之后的 5 个 GDI 函数调用有关。该记录在偏移量 0x0088 处有一个值为 0x0000002B 的类型代码，这代表 EMR_RECTANGLE，很明显是用于 Rectangle 调用的元文件记录。它的长度为 0x00000018（十进制 24）字节，用以容纳 4 个 32 位参数。Rectangle 函数实际上有 5 个参数，但是第一个参数，也就是设备描述表句柄，并未存储在元文件中，因为它没有实际意义。尽管在 EMF2 的函数调用中指定了矩形的顶点坐标分别是(100,100)和(200,200)，但 4 个参数中的两个是 0x00000063 (99)，另外两个是 0x000000C6 (198)。EMF2 程序在 Windows 98 下创建的元文件显示出前两个参数是 0x00000064 (100)，后两个参数是 0x000000C7(199)。显然，在 Rectangle 参数存储到元文件之前，Windows 对它们做了调整，但没有保持一致。这就是对角线端点与矩形顶点不能重合的原因。

其次，有 4 个 16 位记录与 2 个 MoveToEx (0x0000001B 或 EMR_MOVETOEX)和 LineTo (0x00000036 或 EMR_LINETO)调用有关。位于元文件中的参数与传递给函数的参数相同。

元文件以 20 个字节长的类型代码为 0x0000000E 或 EMR_EOF("end of file")的记录结尾。

增强型元文件总是以头记录开始。它对应于 ENHMETAHEADER 类型的结构，定义如下：

```
typedef struct tagENHMETAHEADER
{
    DWORD   iType ;            // EMR_HEADER = 1
    DWORD   nSize ;            // structure size
    RECTL   rclBounds ;        // bounding rectangle in pixels
    RECTL   rclFrame ;         // size of image in 0.01 millimeters
    DWORD   dSignature ;       // ENHMETA_SIGNATURE = " EMF"
```

```
    DWORD   nVersion ;              // 0x00010000
    DWORD   nBytes ;                // file size in bytes
    DWORD   nRecords ;              // total number of records
    WORD    nHandles ;              // number of handles in handle table
    WORD    sReserved ;
    DWORD   nDescription ;          // character length of description string
    DWORD   offDescription ;        // offset of description string in file
    DWORD   nPalEntries ;           // number of entries in palette
    SIZEL   szlDevice ;             // device resolution in pixels
    SIZEL   szlMillimeters ;        // device resolution in millimeters
    DWORD   cbPixelFormat ;         // size of pixel format
    DWORD   offPixelFormat ;        // offset of pixel format
    DWORD   bOpenGL ;               // FALSE if no OpenGL records
}
ENHMETAHEADER ;
```

这种头记录的存在可能是增强型元文件格式对早期 Windows 元文件所做的最为重要的改进。不需要对基于磁盘的元文件使用文件 I/O 函数来获取这些头信息。如果具有元文件句柄，就可以使用 GetEnhMetaFileHeader 函数：

```
GetEnhMetaFileHeader (hemf, cbSize, &emh) ;
```

第一个参数是元文件句柄，最后一个参数是指向 ENHMETAHEADER 结构的指针，第二个参数是该结构的大小。可以使用类似的 GetEnhMetaFileDescription 函数获取描述串。

如上面所定义的，ENHMETAHEADER 结构有 100 字节长，但是在 MF2.EMF 元文件中，记录的大小包括描述串，所以大小为 0x88，即 136 字节。而 Windows 98 元文件的头记录不包含 ENHMETAHEADER 结构的最后 3 个字段，这一点解释了 12 个字节的差别。

rclBounds 字段是指出图像大小的 RECT 结构，单位是像素。将其从十六进制转换过来，我们看到该图像正如我们希望的那样其左上角位于(100,100)，右下角位于(200,200)。

rclFrame 字段是提供相同信息的另一个矩形结构，但它是以 0.01 mm 为单位。在这种情况下，该文件显示两对角顶点分别位于(0x0C35,0x0C35)和(0x186A,0x186A)，用十进制表示为(3125,3125)和(6250,6250)的矩形。这些数字是怎么来的？我们很快就会明白。

dSignature 字段始终为值 ENHMETA_SIGNATURE 或 0x464D4520。这看上去是一个奇怪的数字，但是，如果将字节的排列顺序倒过来(就如同 Intel 处理器在内存中存储多字节数那样)并转换成 ASCII 码，就变成字符串“EMF”。dVersion 字段的值始终是 0x00010000。

其后是 nBytes 字段，该字段在本例中是 0x000000F4，这是该元文件的总字节数。nRecords 字段(在本例中是 0x00000007)指出了记录数——包括头记录、5 个 GDI 函数调用和文件结束记录。

下面是两个 16 位的字段。nHandles 字段为 0x0001。该字段一般指出元文件使用的图形对象(如画笔、画刷和字体)的非默认句柄的数量。由于没有使用这些图形对象,您可能会认为该字段为 0,但实际上 GDI 自己保留了第一个字段。我们将很快见到句柄存储在元文件中的方式。

下两个字段指出描述串的字符个数,以及描述串在文件中的偏移量,这里它们分别为 0x00000012(十进制数 18)和 0x00000064。如果元文件没有描述串,则这两个字段均为 0。

nPalEntries 字段指出在元文件的调色板表中条目的个数,本例中没有这种情况。

头记录接下来是两个 SIZEL 结构,它们包含两个 32 位字段,cx 和 cy。szlDevice 字段(在元文件中的偏移量为 0x0040)指出了以像素为单位的输出设备大小,szlMillimeters 字段(偏移量为 0x0050)指出了以 mm 为单位的输出设备大小。在增强型元文件的参考文档中,这个输出设备被称为"参考设备(reference device)"。它是基于作为第一个参数传递给 CreateEnhMetaFile 调用的句柄所指出的设备描述表的。如果该参数被设置为 NULL,那么 GDI 使用视频显示器。当 EMF2 创建前面所示的元文件时,并且是在 Windows NT 上以 1024 × 768 的视频模式下运行,因此这就是 GDI 被用作参考设备的原因。

GDI 通过调用 GetDeviceCaps 获取此信息。EMF2.EMF 中的 szlDevice 字段是 0x0400 × 0x0300(即 1024 × 768),它是以 HORZRES 和 VERTRES 作为参数调用 GetDeviceCaps 得到的。szlMillimeters 字段是 0x140 × 0xF0(320 × 240),是以 HORZSIZE 和 VERTSIZE 作为参数调用 GetDeviceCaps 得到的。

通过简单的除法就可以得出像素为 0.3125 mm 高和 0.3125 mm 宽,这就是前面描述的 GDI 计算 rclFrame 矩形尺寸的方法。

在元文件中,ENHMETAHEADER 结构后跟一个描述串,该串是 CreateEnhMetaFile 函数的最后一个参数。在本例中,该字符串由后跟一个 NULL 字符的"EMF2"串和后跟两个 NULL 字符的"EMF Demo #2"串组成。总共 18 个字符,要是以 Unicode 方式存储则为 36 个字符。无论创建元文件的程序运行在 Windows NT 还是 Windows 98 下,该字符串始终以 Unicode 方式存储。

18.2.3　元文件和 GDI 对象

我们已经知道了 GDI 绘图命令存储在元文件中的方式,现在看一下 GDI 对象的存储方式。程序 18-4 EMF3 除了创建用于绘制矩形和直线的非默认画笔和画刷以外,与前面介绍的 EMF2 程序很相似。该程序也对 Rectangle 坐标的问题提出了一点修改。EMF3 程序使用 GetVersion 来确定运行环境是 Windows 98 还是 Windows NT,并适当地调整参数。

程序 18-4　EMF3 程序

EMF3.C

```
/*--------------------
   EMF3.C——Enhanced Metafile Demo #3
            (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("EMF3") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #3"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
```

```
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     LOGBRUSH        lb ;
     HDC             hdc, hdcEMF ;
     HENHMETAFILE    hemf ;
     PAINTSTRUCT     ps ;
     RECT            rect ;
     switch (message)
     {
     case WM_CREATE:
          hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf3.emf"), NULL,
                                      TEXT ("EMF3 \0EMF Demo #3 \0")) ;

          SelectObject (hdcEMF, CreateSolidBrush (RGB (0, 0, 255))) ;

          lb.lbStyle = BS_SOLID ;
          lb.lbColor = RGB (255, 0, 0) ;
          lb.lbHatch = 0 ;

          SelectObject (hdcEMF,
               ExtCreatePen (PS_SOLID | PS_GEOMETRIC, 5, &lb, 0, NULL)) ;

          if (GetVersion () & 0x80000000)              // Windows 98
               Rectangle (hdcEMF, 100, 100, 201, 201) ;
          else                                         // Windows NT
               Rectangle (hdcEMF, 101, 101, 202, 202) ;

          MoveToEx  (hdcEMF, 100, 100, NULL) ;
          LineTo    (hdcEMF, 200, 200) ;

          MoveToEx  (hdcEMF, 200, 100, NULL) ;
          LineTo    (hdcEMF, 100, 200) ;

          DeleteObject (SelectObject (hdcEMF, GetStockObject (BLACK_PEN))) ;
          DeleteObject (SelectObject (hdcEMF, GetStockObject (WHITE_BRUSH))) ;

          hemf = CloseEnhMetaFile (hdcEMF) ;
```

```
            DeleteEnhMetaFile (hemf) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;

            rect.left   =     rect.right / 4 ;
            rect.right  = 3 * rect.right / 4 ;
            rect.top    =     rect.bottom / 4 ;
            rect.bottom = 3 * rect.bottom / 4 ;

            hemf = GetEnhMetaFile (TEXT ("emf3.emf")) ;

            PlayEnhMetaFile (hdc, hemf, &rect) ;
            DeleteEnhMetaFile (hemf) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

如我们所看到的，当利用 CreateEnhMetaFile 返回的设备描述表句柄来调用 GDI 函数时，这些函数调用被存储在元文件中，而不是直接输出到屏幕或打印机上。然而，一些 GDI 函数根本不涉及特定的设备描述表。其中有关创建画笔和画刷等图形对象的 GDI 函数十分重要。虽然逻辑画笔和画刷的定义存储在由 GDI 保留的内存中，但是在创建这些对象时，这些抽象的定义并未与任何特定的设备描述表相关。

EMF3 调用 CreateSolidBrush 和 ExtCreatePen 函数。因为这些函数不需要设备描述表句柄，所以 GDI 不会把这些调用存储在元文件里。当调用它们时，GDI 函数只是简单地创建图形绘制对象，而不会影响元文件。

然而，当程序调用 SelectObject 函数将 GDI 对象选入元文件设备描述表时，GDI 既为对象创建函数编码(源自用于存储对象的内部 GDI 数据)，也为元文件中的 SelectObject 调用编码。为了看清其工作方式，我们来看一下 EMF3.EMF 文件的十六进制代码，如图 18-4 所示。

如果把这个元文件同前面的 EMF2.EMF 文件进行比较，第一个不同点就是 EMF3.EMF 头部分中的 rclBounds 字段。在 EMF2.EMF 中，它指出图像限定在坐标(0x64,0x64)和(0xC8,0xC8)区域内。而在 EMF3.EMF 中，坐标是(0x60,0x60)和(0xCC,0xCC)。这表明使用了较粗的笔。rclFrame 字段(以 0.01 mm 为单位指出图像大小)也受到影响。

```
0000   01 00 00 00 88 00 00 00 60 00 00 00 60 00 00 00   ........`...`...
0010   CC 00 00 00 CC 00 00 00 B8 0B 00 00 B8 0B 00 00   ................
0020   E7 18 00 00 E7 18 00 00 20 45 4D 46 00 00 01 00   ........ EMF....
0030   88 01 00 00 0F 00 00 00 03 00 00 00 12 00 00 00   ................
0040   64 00 00 00 00 00 00 00 00 00 04 00 00 00 03 00   d...............
0050   40 01 00 00 F0 00 00 00 00 00 00 00 00 00 00 00   @...............
0060   00 00 00 00 45 00 4D 00 46 00 33 00 00 00 45 00   ....E.M.F.3...E.
0070   4D 00 46 00 20 00 44 00 65 00 6D 00 6F 00 20 00   M.F. .D.e.m.o. .
0080   23 00 33 00 00 00 00 00 27 00 00 00 18 00 00 00   #.3.....'.......
0090   01 00 00 00 00 00 00 00 00 00 FF 00 00 00 00 00   ................
00A0   25 00 00 00 0C 00 00 00 01 00 00 00 5F 00 00 00   %.........._....
00B0   34 00 00 00 02 00 00 00 34 00 00 00 00 00 00 00   4.......4.......
00C0   34 00 00 00 00 00 00 00 00 00 01 00 05 00 00 00   4...............
00D0   00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00   ................
00E0   25 00 00 00 0C 00 00 00 02 00 00 00 2B 00 00 00   %...........+...
00F0   18 00 00 00 63 00 00 00 63 00 00 00 C6 00 00 00   ....c...c.......
0100   C6 00 00 00 1B 00 00 00 10 00 00 00 64 00 00 00   ............d...
0110   64 00 00 00 36 00 00 00 10 00 00 00 C8 00 00 00   d...6...........
0120   C8 00 00 00 1B 00 00 00 10 00 00 00 C8 00 00 00   ................
0130   64 00 00 00 36 00 00 00 10 00 00 00 64 00 00 00   d...6.......d...
0140   C8 00 00 00 25 00 00 00 0C 00 00 00 07 00 00 80   ....%...........
0150   28 00 00 00 0C 00 00 00 02 00 00 00 25 00 00 00   (...........%...
0160   0C 00 00 00 00 00 00 80 28 00 00 00 0C 00 00 00   ........(.......
0170   01 00 00 00 0E 00 00 00 14 00 00 00 00 00 00 00   ................
0180   10 00 00 00 14 00 00 00                           ........
```

图 18-4　EMF3.EMF 的十六制代码

EMF2.EMF 中的 *n*Bytes 字段(偏移量为 0x0030)显示该元文件长度为 0xFA 字节，EMF3.EMF 中长度为 0x0188 字节。EMF2.EMF 元文件包含 7 个记录(1 个头记录，5 个 GDI 函数调用和 1 个文件结束记录)，但是 EMF3.EMF 文件包含 15 个记录。多出的 8 个记录是两个对象创建函数、4 个对 SelectObject 函数的调用和两个对 DeleteObject 函数的调用。

nHandles 字段(在文件中偏移量为 0x0038)指出 GDI 对象的句柄个数。该字段的值总是比元文件使用的非默认对象数多 1。(Platform SDK 文档解释为"保留了此表中的零索引"。)该字段在 EMF2.EMF 的值为 1，而在 EMF3.EMF 中的值为 3，多出的数指出了画笔和画刷。

让我们跳到文件中偏移量为 0x0088 的地方，即第二个记录(头记录之后的第一个

记录)。记录类型为 0x27,对应常量为 EMR_CREATEBRUSHINDIRECT。该元文件记录用于 CreateBrushIndirect 函数,此函数需要指向 LOGBRUSH 结构的指针作为参数。该记录的长度为 0x18(或 24)字节。

每个被选入元文件设备描述表的非备用 GDI 对象得到一个号码,该号码从 1 开始编号。这在此记录的下 4 个字节中指出,在元文件中的偏移量是 0x0090。此记录下面的 3 个 4 字节字段分别对应 LOGBRUSH 结构的 3 个字段:0x00000000(BS_SOLID 的 lbStyle 字段)、0x00FF0000(lbColor 字段)和 0x00000000(lbHatch 字段)。

下一个记录在 EMF3.EMF 中的偏移量为 0x00A0,记录类型为 0x25,或 EMR_SELECTOBJECT,是用于 SelectObject 调用的元文件记录。该记录的长度为 0x0C(或 12)字节,下一个字段是数字 0x01,指出它是选中的第一个 GDI 对象,这就是逻辑画刷。

EMF3.EMF 中的偏移量 0x00AC 是下一个记录,它的记录类型为 0x5F 或 EMR_EXTCREATEPEN。该记录有 0x34(或 52)个字节。下一个 4 字节字段是 0x02,它表示这是在元文件内使用的第二个非备用 GDI 对象。

EMR_EXTCREATEPEN 记录的下 4 个字段重复记录大小两次,之间用 0 字段隔开:0x34、0x00、0x34 和 0x00。下一个字段是 0x00010000,它是 PS_SOLID (0x00000000)与 PS_GEOMETRIC (0x00010000)组合的画笔风格。接下来是 5 个单元的宽度,紧接着是 ExtCreatePen 中使用的逻辑画刷结构的 3 个字段,后接 0 字段。

如果创建了自定义的扩展画笔风格,EMR_EXTCREATEPEN 记录会超过 52 个字节,这样会影响记录的第二字段及两个重复的大小字段。在描述 LOGBRUSH 结构的 3 个字段后面不会是 0(像在 EMF3.EMF 中那样),而是指出了虚线和空格的数量。这后面接着用于虚线和空格长度的许多字段。

EMF3.EMF 的下一个 12 字节的字段是指出第二个对象(画笔)的另一个 SelectObject 调用。接下来的 5 个记录与 EMF2.EMF 中的一样——一个 0x2B(EMR_RECTANGLE)的记录类型和两组 0x1B (EMR_MOVETOEX)和 0x36 (EMR_LINETO)记录。

这些绘图函数后面跟着两组 0x25(EMR_SELECTOBJECT)和 0x28 (EMR_DELETEOBJECT)的 12 字节记录。选择对象记录具有 0x80000007 和 0x80000000 的参数。在设置高位时,它指出一个备用对象,在此例中是 0x07(对应 BLACK_PEN)和 0x00 (WHITE_BRUSH)。

DeleteObject 调用有 2 和 1 的两个参数,用于在元文件中使用的两个非默认对象。虽然 DeleteObject 函数并不需要设备描述表句柄作为它的第一个参数,但 GDI 显然保留了元文件中使用的被程序删除的对象。

最后,元文件以 0x0E(EMF_EOF)记录结束。

总结一下,每当非默认的 GDI 对象首次被选入元文件设备描述表时,GDI 都会为指

出对象创建函数的记录编码(此例中,为 EMR_CREATEBRUSHINDIRECT 和 EMR_EXTCREATEPEN)。每个对象有从 1 开始排序的唯一数值,此数值由记录的第三个字段指出。跟在此记录后的是引用该数值的 EMR_SELECTOBJECT 记录。以后,将对象选入元文件设备描述表(在中间时期没有被删除),只需要 EMR_SELECTOBJECT 记录。

18.2.4　元文件和位图

现在,让我们做点稍微复杂的事,在元文件设备描述表中绘制一幅位图,如程序18-5所示的 EMF4 进行的演示。

程序 18-5　EMF4 程序

EMF4.C

```
/*-------------------------
   EMF4.C——Enhanced Metafile Demo #4
            (c) Charles Petzold, 1998
   -------------------------*/

#define OEMRESOURCE
#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("EMF4") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
```

```
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #4"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;
    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    BITMAP          bm ;
    HBITMAP         hbm ;
    HDC             hdc, hdcEMF, hdcMem ;
    HENHMETAFILE    hemf ;
    PAINTSTRUCT     ps ;
    RECT            rect ;

    switch (message)
    {
    case WM_CREATE:
        hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf4.emf"), NULL,
                                    TEXT ("EMF4 \0EMF Demo #4 \0")) ;

        hbm = LoadBitmap (NULL, MAKEINTRESOURCE (OBM_CLOSE)) ;

        GetObject (hbm, sizeof (BITMAP), &bm) ;

        hdcMem = CreateCompatibleDC (hdcEMF) ;

        SelectObject (hdcMem, hbm) ;

        StretchBlt (hdcEMF, 100, 100, 100, 100,
                    hdcMem,   0,   0, bm.bmWidth, bm.bmHeight, SRCCOPY) ;

        DeleteDC (hdcMem) ;
```

```
            DeleteObject (hbm) ;

            hemf = CloseEnhMetaFile (hdcEMF) ;

            DeleteEnhMetaFile (hemf) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;

            rect.left   =     rect.right / 4 ;
            rect.right  = 3 * rect.right / 4 ;
            rect.top    =     rect.bottom / 4 ;
            rect.bottom = 3 * rect.bottom / 4 ;

            hemf = GetEnhMetaFile (TEXT ("emf4.emf")) ;

            PlayEnhMetaFile (hdc, hemf, &rect) ;
            DeleteEnhMetaFile (hemf) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

为了方便，EMF4 加载由常量 OEM_CLOSE 指出的系统位图。在设备描述表中显示位图的惯用方法是通过调用 CreateCompatibleDC 来创建与目的设备描述表（此例为元文件设备描述表）兼容的内存设备描述表，然后，通过使用 SelectObject 将位图选入该内存设备描述表，并从内存源设备描述表将 BitBlt 或 StretchBlt 调用到目的设备描述表。结束后，删除内存设备描述表和位图。

您会注意到 EMF4 也调用 GetObject 来确定位图的大小。这对 SelectObject 调用是很必要的。

首先，在元文件中存储这些代码对 GDI 来说是个挑战。根本没有函数引导 StretchBlt 调用涉及元文件设备描述表。因此，让我们通过 EMF4.EMF 来看一看它是如何做的，图 18-5 只显示了一部分。

此元文件只包含 3 个记录——头记录、0x0E54 字节长度的 0x4D（或 EMR_STRETCHBLT）和文件结束记录。

我不解释该记录每个字段的含义，但我会指出其关键部分，以便理解 GDI 把 EMF4.C 中的一系列函数调用转化为单个的元文件记录的方法。

```
0000  01 00 00 00 88 00 00 00 64 00 00 00 64 00 00 00  ........d...d...
0010  C7 00 00 00 C7 00 00 00 35 0C 00 00 35 0C 00 00  ........5...5...
0020  4B 18 00 00 4B 18 00 00 20 45 4D 46 00 00 01 00  K...K... EMF....
0030  F0 0E 00 00 03 00 00 00 01 00 00 00 12 00 00 00  ................
0040  64 00 00 00 00 00 00 00 00 04 00 00 00 03 00 00  d...............
0050  40 01 00 00 F0 00 00 00 00 00 00 00 00 00 00 00  @...............
0060  00 00 00 00 45 00 4D 00 46 00 34 00 00 00 45 00  ....E.M.F.4...E.
0070  4D 00 46 00 20 00 44 00 65 00 6D 00 6F 00 20 00  M.F. .D.e.m.o. .
0080  23 00 34 00 00 00 00 00 4D 00 00 00 54 0E 00 00  #.4.....M...T...
0090  64 00 00 00 64 00 00 00 C7 00 00 00 C7 00 00 00  d...d...........
00A0  64 00 00 00 64 00 00 00 64 00 00 00 64 00 00 00  d...d...d...d...
00B0  20 00 CC 00 00 00 00 00 00 00 00 00 00 00 80 3F  ...............?
00C0  00 00 00 00 00 00 00 00 00 00 80 3F 00 00 00 00  ...........?....
00D0  00 00 00 00 FF FF FF 00 00 00 00 00 6C 00 00 00  ............l...
00E0  28 00 00 00 94 00 00 00 C0 0D 00 00 28 00 00 00  (...........(...
00F0  16 00 00 00 28 00 00 00 28 00 00 00 16 00 00 00  ....(...(.......
0100  01 00 20 00 00 00 00 00 C0 0D 00 00 00 00 00 00  ................
0110  00 00 00 00 00 00 00 00 00 00 00 00 C0 C0 C0 00  ................
0120  C0 C0 C0 00 C0 C0 C0 00 C0 C0 C0 00 C0 C0 C0 00  ................
....
0ED0  C0 C0 C0 00 C0 C0 C0 00 C0 C0 C0 00 0E 00 00 00  ................
0EE0  14 00 00 00 00 00 00 00 10 00 00 00 14 00 00 00  ................
```

图 18-5　EMF4.EMF 的部分十六进制代码

GDI 已经把原始的与设备相关的位图转化为与设备无关的位图(DIB)。整个 DIB 存储在用于计算自身大小的记录中。假设当显示元文件和位图时，GDI 实际上使用 StretchDIBits 函数而不是 StretchBlt。或者，GDI 使用 CreateDIBitmap 把 DIB 转变回与设备相关的位图，然后使用内存设备描述表及 StretchBlt 来显示位图。

EMR_STRETCHBLT 记录开始于元文件的偏移量 0x0088 处。DIB 存储在元文件中，以偏移量 0x00F4 开始，到 0x0EDC 处的记录结尾结束。DIB 以 BITMAPINFOHEADER 类型的 40 字节的结构开始。在偏移量 0x011C 处接有 22 个像素行，每行 40 个像素。这是每像素 32 位的 DIB，所以每个像素需要 4 个字节。

18.2.5　列举元文件

当您希望访问元文件的单独记录时，可以使用被称作元文件列举的过程。如程序 18-6

EMF5 所示。此程序使用元文件来显示与 EMF3 相同的图像，但它是通过使用元文件列举来工作的。

程序 18-6　EMF5 程序

EMF5.C

```
/*-------------------
   EMF5.C——Enhanced Metafile Demo #5
            (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("EMF5") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #5"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
```

```
    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

int CALLBACK EnhMetaFileProc (HDC hdc, HANDLETABLE * pHandleTable,
                              CONST ENHMETARECORD * pEmfRecord,
                              int iHandles, LPARAM pData)
{
    PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfRecord, iHandles) ;

    return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    HDC             hdc ;
    HENHMETAFILE    hemf ;
    PAINTSTRUCT     ps ;
    RECT            rect ;

    switch (message)
    {
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;

        rect.left   =     rect.right / 4 ;
        rect.right  = 3 * rect.right / 4 ;
        rect.top    =     rect.bottom / 4 ;
        rect.bottom = 3 * rect.bottom / 4 ;

        hemf = GetEnhMetaFile (TEXT ("..\\emf3\\emf3.emf")) ;

        EnumEnhMetaFile (hdc, hemf, EnhMetaFileProc, NULL, &rect) ;
        DeleteEnhMetaFile (hemf) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
```

```
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

此程序使用 EMF3 程序创建的 EMF3.EMF 文件，所以确保运行此程序前先运行 EMF3 程序。同时，需要在 Visual C++ 环境中运行两个程序，以确保路径的正确。在处理 WM_PAINT 过程中，两个程序的主要区别是：EMF3 调用 PlayEnhMetaFile，而 EMF5 调用 EnumEnhMetaFile。PlayEnhMetaFile 函数有下面的语法：

```
PlayEnhMetaFile (hdc, hemf, &rect) ;
```

第一个参数是要显示的元文件的设备描述表句柄。第二个参数是增强型元文件句柄。第三个参数是指向描述设备描述表平面上矩形的 RECT 结构的指针。元文件图像伸展以适合但不超出该矩形。

EnumEnhMetaFile 有 5 个参数，其中 3 个与 PlayEnhMetaFile 一样（虽然 RECT 结构的指针已经移到参数表的末尾）。

EnumEnhMetaFile 的第三个参数是列举函数的名称，它用于调用 EnhMetaFileProc。第四个参数是希望传递给列举函数的任意数据的指针，这里将该参数简单地设置为 NULL。

现在看一看列举函数。当调用 EnumEnhMetaFile 时，对于元文件中的每一个记录，GDI 都将调用 EnhMetaFileProc 一次，包括头记录和文件结束记录。通常列举函数返回 TRUE，但它可能返回 FALSE 以忽略列举过程。

该列举函数有 5 个参数，稍后会描述它们。在这个程序中，我仅把前 4 个参数传递给 PlayEnhMetaFileRecord，它使 GDI 执行由该记录代表的函数调用，好像您明确地调用它一样。

EMF5 使用 EnumEnhMetaFile 和 PlayEnhMetaFileRecord 得到的结果与 EMF3 调用 PlayEnhMetaFile 得到的结果一样。区别在于 EMF5 有一个元文件提交过程的连接，并能够访问每个元文件记录。这是很有用的。

列举函数的第一个参数是设备描述表句柄。GDI 从 EnumEnhMetaFile 的第一个参数中简单地获取此句柄。列举函数把该句柄传递给 PlayEnhMetaFileRecord 来标识图像提交到的设备描述表。

我们先跳到列举函数的第三个参数，它是指向 ENHMETARECORD 类型结构的指针，前面已经提到过。这个结构描述实际的元文件记录，就像它亲自在元文件中编码一样。

您可以写一些代码来检查这些记录。您也许不想把某些记录传送到 PlayEnhMetaFileRecord 函数。例如，在 EMF5.C 中，把下行插入到 PlayEnhMetaFileRecord 调用的前面：

```
if (pEmfRecord -> iType != EMR_LINETO)
```

重新编译该程序，运行它，将只看到矩形，而没有两条线。或使用下面的语句：

```
if (pEmfRecord -> iType != EMR_SELECTOBJECT)
```

这个小变化会导致用默认对象提交图像，而不是用创建的画笔和画刷。

不应做的一件事是修改元文件记录，先不要对此太担心，让我们看一看程序 18-7 EMF6。

程序 18-7　EMF6 程序

EMF6.C

```
/*-------------------
   EMF6.C——Enhanced Metafile Demo #6
            (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR lpszCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("EMF6") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
```

```
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #6"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;
     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

int CALLBACK EnhMetaFileProc (HDC hdc, HANDLETABLE * pHandleTable,
                              CONST ENHMETARECORD * pEmfRecord,
                              int iHandles, LPARAM pData)
{
     ENHMETARECORD * pEmfr ;

     pEmfr = (ENHMETARECORD *) malloc (pEmfRecord -> nSize) ;

     CopyMemory (pEmfr, pEmfRecord, pEmfRecord -> nSize) ;

     if (pEmfr -> iType == EMR_RECTANGLE)
          pEmfr -> iType = EMR_ELLIPSE ;

     PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfr, iHandles) ;

     free (pEmfr) ;

     return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     HDC             hdc ;
     HENHMETAFILE    hemf ;
     PAINTSTRUCT     ps ;
     RECT            rect ;

     switch (message)
     {
     case WM_PAINT:
```

```
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;

        rect.left   =     rect.right / 4 ;
        rect.right  = 3 * rect.right / 4 ;
        rect.top    =     rect.bottom / 4 ;
        rect.bottom = 3 * rect.bottom / 4 ;

        hemf = GetEnhMetaFile (TEXT ("..\\emf3\\emf3.emf")) ;

        EnumEnhMetaFile (hdc, hemf, EnhMetaFileProc, NULL, &rect) ;
        DeleteEnhMetaFile (hemf) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

与 EMF5 一样，EMF6 使用 EMF3 程序创建的 EMF3.EMF 元文件，因此确保在此之前运行 EMF3 程序，并且在 Visual C++ 中运行所有程序。

EMF6 演示了如果在提交元文件之前要修改它们，解决方法是非常简单的：做个副本并修改它。可以看到，列举过程一开始使用 malloc 分配一块元文件记录大小的内存，它是由传递给该函数的 pEmfRecord 结构的 nSize 字段指出的。这个块的指针保存在变量 pEmfr 中，pEmfr 本身是指向 ENHMETARECORD 结构的指针。

程序使用 CopyMemory 把 pEmfRecord 指向的结构内容复制到 pEmfr 指向的结构中。现在我们就可以做出一些改变。程序检查记录是否为 EMR_RECTANGLE 类型，如果是，则用 EMR_ELLIPSE 取代 iType 字段。PEmfr 指针被传递到 PlayEnhMetaFileRecord，然后被释放。结果是程序画出一个椭圆而不是矩形，其他的事是相同的。

当然，我们的小改变很容易起作用，因为 Rectangle 和 Ellipse 函数有同样的参数，这些参数都定义同一件事——图画的边界框。要进行范围更广的修改，需要一些有关不同元文件记录格式的知识。

另一个可能性是插入一两个额外的记录。例如，用下面的语句代替 EMF6.C 中的 if 语句：

```
if (pEmfr -> iType == EMR_RECTANGLE)
{
    PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfr, nObjects) ;
```

```
    pEmfr -> iType = EMR_ELLIPSE ;
}
```

无论何时出现 Rectangle 记录，程序都提交此记录，并把它更改为 Ellipse，然后再提交。现在程序将画出矩形和椭圆。

现在讨论一下当列举元文件时 GDI 对象处理的方式。

在元文件头中，ENHMETAHEADER 结构的 nHandles 字段是比在元文件中创建的 GDI 对象数大的值。因此，对于 EMF5 和 EMF6 中的元文件，此字段是 3，表明画笔、画刷和其他东西。“其他东西”的具体内容，稍后我会说明。

您会注意到 EMF5 和 EMF6 中列举函数的倒数第二个参数，也称作 nHandles，它是同一个数 3。

列举函数的第二个参数是指向 HANDLETABLE 结构的指针，在 WINGDI.H 中定义如下：

```
typedef struct tagHANDLETABLE
{
    HGDIOBJ objectHandle [1] ;
}
HANDLETABLE ;
```

HGDIOBJ 数据类型是 GDI 对象的句柄，被定义为 32 位的指针，类似于所有其他 GDI 对象。这是那些带有一个元素的数组字段的结构之一。这意味着此字段具有可变的长度。objectHandle 数组中的元素数等于 nHandles，在此程序中是 3。

在列举函数中，可以使用以下表达式获取这些句柄：

```
pHandleTable -> objectHandle[i]
```

对于 3 个句柄，i 是 0、1 和 2。

每当调用列举函数时，数组的第一个元素都将包含所列举的元文件句柄。这就是前面提到的“其他东西”。

当第一次调用列举函数时，表的第二和第三个元素将是 0。它们是画笔和画刷句柄的保留位置。

以下是工作方式：元文件中的第一个对象创建函数具有 EMR_CREATEBRUSHINDIRECT 的记录类型，此记录指出了对象号 1。当把该记录传递给 PlayEnhMetaFileRecord 时，GDI 创建画刷并获取它的句柄。此句柄存储在 objectHandle 数组的元素 1（第二个元素）中。当把第一个 EMR_SELECTOBJECT 记录传递给 PlayEnhMetaFileRecord 时，GDI 发现此句柄为 1，并能从表中检索到实际的句柄并把它用于 SelectObject 调用。当元文件最终删除画刷时，GDI 将 objectHandle 数组的元素 1 设置回 0。

通过访问 objectHandle 数组，可以使用例如 GetObjectType 和 GetObject 等调用获取在元文件中使用的对象信息。

18.2.6 嵌入图像

对元文件列举的最重要的应用也许是在现有的元文件中嵌入其他图像(甚至是整个元文件)。事实上,现有的元文件保持不变,真正做的是创建包含现有元文件和新嵌入图像的新元文件。基本的技巧是把元文件设备描述表句柄传递给 EnumEnhMetaFile,作为它的第一个参数。这使得您能够在元文件设备描述表上提交元文件记录和 GDI 函数调用。

在元文件命令序列的开头或结尾嵌入新图像是极简单的——正好在 EMR_HEADER 记录之后或在 EMF_EOF 记录之前。然而,如果您熟悉现有的元文件,就可以把新的绘图命令嵌入所需的任何地方,如程序 18-8 所示的 EMF7 进行的演示。

程序 18-8　EMF7 程序

EMF7.C

```
/*-------------------------
    EMF7.C——Enhanced Metafile Demo #7
             (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR lpszCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("EMF7") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;
```

```
    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Demo #7"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

int CALLBACK EnhMetaFileProc (HDC hdc, HANDLETABLE * pHandleTable,
                              CONST ENHMETARECORD * pEmfRecord,
                              int iHandles, LPARAM pData)
{
    HBRUSH      hBrush ;
    HPEN        hPen ;
    LOGBRUSH    lb ;

    if (pEmfRecord -> iType != EMR_HEADER && pEmfRecord -> iType != EMR_EOF)

        PlayEnhMetaFileRecord (hdc, pHandleTable, pEmfRecord, iHandles) ;

    if (pEmfRecord -> iType == EMR_RECTANGLE)
    {
        hBrush = SelectObject (hdc, GetStockObject (NULL_BRUSH)) ;

        lb.lbStyle = BS_SOLID ;
        lb.lbColor = RGB (0, 255, 0) ;
        lb.lbHatch = 0 ;

        hPen = SelectObject (hdc,
            ExtCreatePen (PS_SOLID | PS_GEOMETRIC, 5, &lb, 0, NULL)) ;
```

```
            Ellipse (hdc, 100, 100, 200, 200) ;

            DeleteObject (SelectObject (hdc, hPen)) ;
            SelectObject (hdc, hBrush) ;
        }
        return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        ENHMETAHEADER  emh ;
        HDC            hdc, hdcEMF ;
        HENHMETAFILE   hemfOld, hemf ;
        PAINTSTRUCT    ps ;
        RECT           rect ;

        switch (message)
        {
        case WM_CREATE:

                // Retrieve existing metafile and header

            hemfOld = GetEnhMetaFile (TEXT ("..\\emf3\\emf3.emf")) ;

            GetEnhMetaFileHeader (hemfOld, sizeof (ENHMETAHEADER), &emh) ;

                // Create a new metafile DC

            hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf7.emf"), NULL,
                                        TEXT ("EMF7\0EMF Demo #7\0")) ;

                // Enumerate the existing metafile

            EnumEnhMetaFile (hdcEMF, hemfOld, EnhMetaFileProc, NULL,
                             (RECT *) &emh.rclBounds) ;

                // Clean up

            hemf = CloseEnhMetaFile (hdcEMF) ;

            DeleteEnhMetaFile (hemfOld) ;
            DeleteEnhMetaFile (hemf) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;
```

```
            rect.left   =     rect.right / 4 ;
            rect.right  = 3 * rect.right / 4 ;
            rect.top    =     rect.bottom / 4 ;
            rect.bottom = 3 * rect.bottom / 4 ;

            hemf = GetEnhMetaFile (TEXT ("emf7.emf")) ;

            PlayEnhMetaFile (hdc, hemf, &rect) ;
            DeleteEnhMetaFile (hemf) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

EMF7 使用 EMF3 程序创建的 EMF3.EMF 元文件，所以在运行 EMF7 之前要运行 EMF3 程序创建元文件。

EMF7 中的 WM_PAINT 处理使用 PlayEnhMetaFile 而不是 EnumEnhMetaFile，而且 WM_CREATE 处理有很大的差别。

首先，程序通过调用 GetEnhMetaFile 获取 EMF3.EMF 文件的元文件句柄，还调用 GetEnhMetaFileHeader 得到增强型元文件头记录，目的是在后面的 EnumEnhMetaFile 调用中使用 rclBounds 字段。

接下来，程序创建新的基于磁盘的元文件，名为 EMF7.EMF。CreateEnhMetaFile 函数为元文件返回设备描述表句柄。然后，使用 EMF7.EMF 的元文件设备描述表句柄和 EMF3.EMF 的元文件句柄调用 EnumEnhMetaFile。

现在让我们来看一看 EnhMetaFileProc。如果被列举的记录不是头记录，也不是文件结束记录，那么函数就调用 PlayEnhMetaFileRecord 把记录转换为新的元文件设备描述表。（排除头记录或文件结束记录并不是十分有必要，但是它们可以使元文件增大一些。）

如果刚转换的记录是 Rectangle 调用，则函数创建画笔来绘制具有绿色的轮廓线和透明内部椭圆。注意代码通过保存先前的画笔和画刷句柄来恢复设备描述表状态的方法。在此期间，所有这些函数都被插入到元文件中。（记住，也可以使用 PlayEnhMetaFile 在现有的元文件中插入整个元文件。）

回到 WM_CREATE 处理，程序调用 CloseEnhMetaFile 获取新元文件的句柄。然后，它删除两个元文件句柄，将 EMF3.EMF 和 EMF7.EMF 文件保留在磁盘上。

从程序显示输出中可以很明显地看到，该椭圆是在矩形之后、两条对角线之前绘制

的。

18.2.7 增强型元文件查看器和打印机

使用剪贴板转换增强型元文件非常简单，剪贴板类型是 CF_ENHMETAFILE。GetClipboardData 函数返回增强型元文件句柄，SetClipboardData 也使用该元文件句柄。复制元文件时可以使用 CopyEnhMetaFile 函数。如果把增强型元文件放在剪贴板中，Windows 为需要它的那些程序使元文件在老式的格式下也可用。如果在剪贴板中放置老式格式的元文件，Windows 将使增强型元文件也可用。

程序 18-9 所示的 EMFVIEW 显示了在剪贴板中转移元文件的代码，它也允许加载、存储和打印元文件。

程序 18-9　EMFVIEW 程序

EMFVIEW.C

```
/* - - - - - - - - - - - - - - - - - - - -
    EMFVIEW.C——View Enhanced Metafiles
                (c) Charles Petzold, 1998
   - - - - - - - - - - - - - - - - - - - - */

# include <windows.h>
# include <commdlg.h>
# include "resource.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("EmfView") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    HACCEL     hAccel ;
    HWND       hwnd ;
    MSG        msg ;
    WNDCLASS   wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
```

```
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Enhanced Metafile Viewer"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     hAccel = LoadAccelerators (hInstance, szAppName) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          if (!TranslateAccelerator (hwnd, hAccel, &msg))
          {
               TranslateMessage (&msg) ;
               DispatchMessage (&msg) ;
          }
     }
     return msg.wParam ;
}

HPALETTE CreatePaletteFromMetaFile (HENHMETAFILE hemf)
{
     HPALETTE        hPalette ;
     int             iNum ;
     LOGPALETTE *    plp ;

     if (!hemf)
          return NULL ;

     if (0 == (iNum = GetEnhMetaFilePaletteEntries (hemf, 0, NULL)))
          return NULL ;

     plp = malloc (sizeof (LOGPALETTE) + (iNum - 1) * sizeof (PALETTEENTRY)) ;

     plp -> palVersion    = 0x0300 ;
     plp -> palNumEntries = iNum ;
```

```
    GetEnhMetaFilePaletteEntries (hemf, iNum, plp -> palPalEntry) ;

    hPalette = CreatePalette (plp) ;

    free (plp) ;

    return hPalette ;

}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static DOCINFO          di = { sizeof (DOCINFO), TEXT ("EmfView: Printing") } ;
    static HENHMETAFILE     hemf ;
    static OPENFILENAME     ofn ;
    static PRINTDLG         printdlg = { sizeof (PRINTDLG) } ;
    static TCHAR            szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
    static TCHAR            szFilter[] =
                                TEXT ("Enhanced Metafiles (*.EMF)\0*.emf\0")
                                TEXT ("All Files (*.*)\0*.*\0\0") ;
    BOOL                    bSuccess ;
    ENHMETAHEADER           header ;
    HDC                     hdc, hdcPrn ;
    HENHMETAFILE            hemfCopy ;
    HMENU                   hMenu ;
    HPALETTE                hPalette ;
    int                     i, iLength, iEnable ;
    PAINTSTRUCT             ps ;
    RECT                    rect ;
    PTSTR                   pBuffer ;

    switch (message)
    {
    case WM_CREATE:
            // Initialize OPENFILENAME structure

        ofn.lStructSize       = sizeof (OPENFILENAME) ;
        ofn.hwndOwner         = hwnd ;
        ofn.hInstance         = NULL ;
        ofn.lpstrFilter       = szFilter ;
        ofn.lpstrCustomFilter = NULL ;
        ofn.nMaxCustFilter    = 0 ;
        ofn.nFilterIndex      = 0 ;
        ofn.lpstrFile         = szFileName ;
        ofn.nMaxFile          = MAX_PATH ;
        ofn.lpstrFileTitle    = szTitleName ;
        ofn.nMaxFileTitle     = MAX_PATH ;
```

```
        ofn.lpstrInitialDir     = NULL ;
        ofn.lpstrTitle          = NULL ;
        ofn.Flags               = 0 ;
        ofn.nFileOffset         = 0 ;
        ofn.nFileExtension      = 0 ;
        ofn.lpstrDefExt         = TEXT ("emf") ;
        ofn.lCustData           = 0 ;
        ofn.lpfnHook            = NULL ;
        ofn.lpTemplateName      = NULL ;
        return 0 ;

    case WM_INITMENUPOPUP:
        hMenu = GetMenu (hwnd) ;

        iEnable = hemf ? MF_ENABLED : MF_GRAYED ;

        EnableMenuItem (hMenu, IDM_FILE_SAVE_AS,      iEnable) ;
        EnableMenuItem (hMenu, IDM_FILE_PRINT,        iEnable) ;
        EnableMenuItem (hMenu, IDM_FILE_PROPERTIES,   iEnable) ;
        EnableMenuItem (hMenu, IDM_EDIT_CUT,          iEnable) ;
        EnableMenuItem (hMenu, IDM_EDIT_COPY,         iEnable) ;
        EnableMenuItem (hMenu, IDM_EDIT_DELETE,       iEnable) ;

        EnableMenuItem (hMenu, IDM_EDIT_PASTE,
            IsClipboardFormatAvailable (CF_ENHMETAFILE) ?
                MF_ENABLED : MF_GRAYED) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_OPEN:
                // Show the File Open dialog box

            ofn.Flags = 0 ;

            if (!GetOpenFileName (&ofn))
                return 0 ;

                // If there's an existing EMF, get rid of it.

            if (hemf)
            {
                DeleteEnhMetaFile (hemf) ;
                hemf = NULL ;
            }
                // Load the EMF into memory
```

```
            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            hemf = GetEnhMetaFile (szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                // Invalidate the client area for later update

            InvalidateRect (hwnd, NULL, TRUE) ;

            if (hemf == NULL)
            {
                MessageBox (hwnd, TEXT ("Cannot load metafile"),
                            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
            }
            return 0 ;

        case IDM_FILE_SAVE_AS:
            if (!hemf)
                return 0 ;

                // Show the File Save dialog box

            ofn.Flags = OFN_OVERWRITEPROMPT ;

            if (!GetSaveFileName (&ofn))
                return 0 ;

                // Save the EMF to disk file

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            hemfCopy = CopyEnhMetaFile (hemf, szFileName) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
            if (hemfCopy)
            {
                DeleteEnhMetaFile (hemf) ;
                hemf = hemfCopy ;
            }
            else
                MessageBox (hwnd, TEXT ("Cannot save metafile"),
                            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
```

```
            return 0 ;

        case IDM_FILE_PRINT:
                // Show the Print dialog box and get printer DC

            printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;

            if (!PrintDlg (&printdlg))
                return 0 ;

            if (NULL == (hdcPrn = printdlg.hDC))
            {
                 MessageBox (hwnd, TEXT ("Cannot obtain printer DC"),
                             szAppName, MB_ICONEXCLAMATION | MB_OK) ;
                 return 0 ;
            }
                // Get size of printable area of page

            rect.left   = 0 ;
            rect.right  = GetDeviceCaps (hdcPrn, HORZRES) ;
            rect.top    = 0 ;
            rect.bottom = GetDeviceCaps (hdcPrn, VERTRES) ;

            bSuccess = FALSE ;

                // Play the EMF to the printer

            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            if ((StartDoc (hdcPrn, &di) > 0) && (StartPage (hdcPrn) > 0))
            {
                 PlayEnhMetaFile (hdcPrn, hemf, &rect) ;

                 if (EndPage (hdcPrn) > 0)
                 {
                      bSuccess = TRUE ;
                      EndDoc (hdcPrn) ;
                 }
            }
            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            DeleteDC (hdcPrn) ;

            if (!bSuccess)
                MessageBox (hwnd, TEXT ("Could not print metafile"),
                            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
```

```
            return 0 ;

        case IDM_FILE_PROPERTIES:
            if (!hemf)
                return 0 ;

            iLength = GetEnhMetaFileDescription (hemf, 0, NULL) ;
            pBuffer = malloc ((iLength + 256) * sizeof (TCHAR)) ;

            GetEnhMetaFileHeader (hemf, sizeof (ENHMETAHEADER), &header) ;

                // Format header file information

            i = wsprintf (pBuffer,
                          TEXT ("Bounds = (%i, %i) to (%i, %i) pixels \n"),
                          header.rclBounds.left, header.rclBounds.top,
                          header.rclBounds.right, header.rclBounds.bottom) ;

            i += wsprintf (pBuffer + i,
                           TEXT ("Frame = (%i, %i) to (%i, %i) mms \n"),
                           header.rclFrame.left, header.rclFrame.top,
                           header.rclFrame.right, header.rclFrame.bottom) ;

            i += wsprintf (pBuffer + i,
                           TEXT ("Resolution = (%i, %i) pixels")
                           TEXT (" = (%i, %i) mms \n"),
                           header.szlDevice.cx, header.szlDevice.cy,
                           header.szlMillimeters.cx,
                           header.szlMillimeters.cy) ;

            i += wsprintf (pBuffer + i,
                           TEXT ("Size = %i, Records = %i, ")
                           TEXT ("Handles = %i, Palette entries = %i \n"),
                           header.nBytes, header.nRecords,
                           header.nHandles, header.nPalEntries) ;
                // Include the metafile description, if present

            if (iLength)
            {
                i += wsprintf (pBuffer + i, TEXT ("Description = ")) ;
                GetEnhMetaFileDescription (hemf, iLength, pBuffer + i) ;
                pBuffer [lstrlen (pBuffer)] = '\t' ;
            }

            MessageBox (hwnd, pBuffer, TEXT ("Metafile Properties"), MB_OK) ;
            free (pBuffer) ;
            return 0 ;
```

```
case IDM_EDIT_COPY:
case IDM_EDIT_CUT:
     if (!hemf)
          return 0 ;

          // Transfer metafile copy to the clipboard

     hemfCopy = CopyEnhMetaFile (hemf, NULL) ;

     OpenClipboard (hwnd) ;
     EmptyClipboard () ;
     SetClipboardData (CF_ENHMETAFILE, hemfCopy) ;
     CloseClipboard () ;

     if (LOWORD (wParam) == IDM_EDIT_COPY)
          return 0 ;
                              // fall through if IDM_EDIT_CUT

case IDM_EDIT_DELETE:
     if (hemf)
     {
          DeleteEnhMetaFile (hemf) ;
          hemf = NULL ;
          InvalidateRect (hwnd, NULL, TRUE) ;
     }
     return 0 ;

case IDM_EDIT_PASTE:
     OpenClipboard (hwnd) ;
     hemfCopy = GetClipboardData (CF_ENHMETAFILE) ;

     CloseClipboard () ;
     if (hemfCopy && hemf)
     {
          DeleteEnhMetaFile (hemf) ;
          hemf = NULL ;
     }

     hemf = CopyEnhMetaFile (hemfCopy, NULL) ;
     InvalidateRect (hwnd, NULL, TRUE) ;
     return 0 ;

case IDM_APP_ABOUT:
     MessageBox (hwnd, TEXT ("Enhanced Metafile Viewer \ n")
                       TEXT ("(c) Charles Petzold, 1998"),
                 szAppName, MB_OK) ;
     return 0 ;
```

```
        case IDM_APP_EXIT:
            SendMessage (hwnd, WM_CLOSE, 0, 0L) ;
            return 0 ;
        }
        break ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hemf)
        {
            if (hPalette = CreatePaletteFromMetaFile (hemf))

                SelectPalette (hdc, hPalette, FALSE) ;
                RealizePalette (hdc) ;

            GetClientRect (hwnd, &rect) ;
            PlayEnhMetaFile (hdc, hemf, &rect) ;

            if (hPalette)
                DeleteObject (hPalette) ;
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_QUERYNEWPALETTE:
        if (!hemf || !(hPalette = CreatePaletteFromMetaFile (hemf)))
            return FALSE ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
        InvalidateRect (hwnd, NULL, FALSE) ;

        DeleteObject (hPalette) ;
        ReleaseDC (hwnd, hdc) ;
        return TRUE ;

    case WM_PALETTECHANGED:
        if ((HWND) wParam == hwnd)
            break ;

        if (!hemf || !(hPalette = CreatePaletteFromMetaFile (hemf)))
            break ;

        hdc = GetDC (hwnd) ;
        SelectPalette (hdc, hPalette, FALSE) ;
        RealizePalette (hdc) ;
```

```
            UpdateColors (hdc) ;

            DeleteObject (hPalette) ;
            ReleaseDC (hwnd, hdc) ;
            break ;

        case WM_DESTROY:
            if (hemf)
                DeleteEnhMetaFile (hemf) ;

            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

EMFVIEW.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

EMFVIEW MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&Open \tCtrl+O",              IDM_FILE_OPEN
        MENUITEM "Save &As...",                 IDM_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "&Print...\tCtrl+P",           IDM_FILE_PRINT
        MENUITEM SEPARATOR
        MENUITEM "&Properties",                 IDM_FILE_PROPERTIES
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       IDM_APP_EXIT
    END
    POPUP "&Edit"
    BEGIN
        MENUITEM "Cu&t\tCtrl+X",                IDM_EDIT_CUT
        MENUITEM "&Copy\tCtrl+C",               IDM_EDIT_COPY
        MENUITEM "&Paste\tCtrl+V",              IDM_EDIT_PASTE
        MENUITEM "&Delete\tDel",                IDM_EDIT_DELETE
    END
    POPUP "Help"
    BEGIN
```

```
        MENUITEM "&About EmfView...",              IDM_APP_ABOUT
    END
END

//////////////////////////////////////////////////////////////////////////////
// Accelerator

EMFVIEW ACCELERATORS DISCARDABLE
BEGIN
    "C",            IDM_EDIT_COPY,      VIRTKEY, CONTROL, NOINVERT
    "O",            IDM_FILE_OPEN,      VIRTKEY, CONTROL, NOINVERT
    "P",            IDM_FILE_PRINT,     VIRTKEY, CONTROL, NOINVERT
    "V",            IDM_EDIT_PASTE,     VIRTKEY, CONTROL, NOINVERT
    VK_DELETE,      IDM_EDIT_DELETE,    VIRTKEY, NOINVERT
    "X",            IDM_EDIT_CUT,       VIRTKEY, CONTROL, NOINVERT
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by EmfView.rc

#define IDM_FILE_OPEN               40001
#define IDM_FILE_SAVE_AS            40002
#define IDM_FILE_PRINT              40003
#define IDM_FILE_PROPERTIES         40004
#define IDM_APP_EXIT                40005
#define IDM_EDIT_CUT                40006
#define IDM_EDIT_COPY               40007
#define IDM_EDIT_PASTE              40008
#define IDM_EDIT_DELETE             40009
#define IDM_APP_ABOUT               40010
```

EMFVIEW 也具有完整的调色板逻辑，只有在元文件中对调色板编码才能做到这一点。（通过调用 SelectPalette 实现。）该程序在 CreatePaletteFromMetaFile 函数中处理调色板，在处理 WM_PAINT 显示元文件以及处理 WM_QUERYNEWPALETTE 和 WM_PALETTECHANGED 消息时，调用这个函数。

在响应菜单中的 Print 命令时，EMFVIEW 显示普通的打印机对话框，然后获取页面中可打印区域的大小。元文件伸展以填充整个区域。EMFVIEW 在窗口中类似地显示元文件。

File 菜单中的 Properties 选项使 EMFVIEW 显示包含元文件头信息的消息框。

如果打印本章前面创建的 EMF2.EMF 元文件图像，您将会发现用高分辨率的打印机打印出的线条非常细，几乎看不清。打印矢量图像时应该使用较宽的画笔（例如，一

磅宽)。本章后面所示的直尺图像就是这样做的。

18.2.8　显示精确的元文件图像

元文件图像的好处在于它能够以任意大小缩放并且仍能保持一定的逼真度。这是因为元文件通常由一系列矢量图形的图元组成,图元是指线条、填充的区域,以及轮廓字体等。扩大或缩小图像只是简单地缩放定义这些图元的所有坐标点。另一方面,对位图来说,压缩图像会丢掉整行和整列像素,这样就丢失了重要的信息。

当然,元文件的压缩并不是完美无缺的。我们所使用的图形输出设备的像素大小是有限的。当元文件图像压缩到一定大小时,组成元文件的大量线条会变成模糊的斑点,同时区域填充图案和混合色看起来也很奇怪。如果元文件中包含嵌入的位图或老式的点阵字体,同样会引起类似的问题。

尽管如此,大多数情况下元文件可以任意地缩放。这在把元文件放入字处理或桌面印刷文档中时非常有用。一般情况下,在上述的应用程序中选择元文件图像时,会出现围绕图像的矩形,您可以用鼠标拖动该矩形,将它缩放为任意大小。图像提交到打印机时,它也具有同样的相对尺寸。

然而,有时任意缩放元文件并不是个好主意。例如,假设您有一个存储着储户签名样本的银行系统,这些签名作为一系列折线存储在元文件中。将元文件变宽或变高会使签名看起来不同,因此应该保持图像的纵横比一致。

在前面的示例程序中,是以客户区的大小来确定 PlayEnhMetaFile 调用中的围绕矩形。于是,如果改变程序窗体的大小,也就改变了图像的大小。这与在字处理文档中改变元文件图像大小的概念相似。

正确地显示元文件图像(以特定的度量单位或用适当的纵横比),需要使用元文件头中的大小信息并根据此信息设置矩形结构。

在本章余下的示例程序中将使用名为 EMF.C 的外壳程序,它包括打印逻辑、资源描述文件 EMF.RC 和头文件 RESOURCE.H。程序 18-10 显示了这些文件及 EMF8.C 程序,该程序使用这些文件显示一把 6 英寸的直尺。

程序 18-10　EMF8 程序

EMF8.C

```
/*-------------------------
   EMF8.C——Enhanced Metafile Demo #8
            (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
```

```
TCHAR szClass [ ] = TEXT ("EMF8") ;
TCHAR szTitle [ ] = TEXT ("EMF8: Enhanced Metafile Demo #8") ;

void DrawRuler (HDC hdc, int cx, int cy)
{
     int       iAdj, i, iHeight ;
     LOGFONTlf ;
     TCHAR  ch ;

     iAdj = GetVersion () & 0x80000000 ? 0 : 1 ;
          // Black pen with 1-point width

     SelectObject (hdc, CreatePen (PS_SOLID, cx / 72 / 6, 0)) ;

          // Rectangle surrounding entire pen (with adjustment)

     Rectangle (hdc, iAdj, iAdj, cx + iAdj + 1, cy + iAdj + 1) ;

          // Tick marks

     for (i = 1 ; i < 96 ; i++)
     {
               if (i % 16 == 0) iHeight = cy /  2 ;     // inches
          else if (i %  8 == 0) iHeight = cy /  3 ;     // half inches
          else if (i %  4 == 0) iHeight = cy /  5 ;     // quarter inches
          else if (i %  2 == 0) iHeight = cy /  8 ;     // eighths
          else                  iHeight = cy / 12 ;     // sixteenths

          MoveToEx (hdc, i * cx / 96, cy, NULL) ;
          LineTo    (hdc, i * cx / 96, cy - iHeight) ;
     }
          // Create logical font

     FillMemory (&lf, sizeof (lf), 0) ;
     lf.lfHeight = cy / 2 ;
     lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;

     SelectObject (hdc, CreateFontIndirect (&lf)) ;
     SetTextAlign (hdc, TA_BOTTOM | TA_CENTER) ;
     SetBkMode  (hdc, TRANSPARENT) ;

          // Display numbers

     for (i = 1 ; i <= 5 ; i++)
     {
          ch = (TCHAR) (i + '0') ;
          TextOut (hdc, i * cx / 6, cy / 2, &ch, 1) ;
     }
```

```
        // Clean up

    DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
    DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN))) ;
}

void CreateRoutine (HWND hwnd)
{
    HDC             hdcEMF ;
    HENHMETAFILE    hemf ;
    int             cxMms, cyMms, cxPix, cyPix, xDpi, yDpi ;

    hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf8.emf"), NULL,
                                TEXT ("EMF8 \ 0EMF Demo # 8 \ 0")) ;
    if (hdcEMF == NULL)
        return ;

    cxMms = GetDeviceCaps (hdcEMF, HORZSIZE) ;
    cyMms = GetDeviceCaps (hdcEMF, VERTSIZE) ;
    cxPix = GetDeviceCaps (hdcEMF, HORZRES) ;
    cyPix = GetDeviceCaps (hdcEMF, VERTRES) ;

    xDpi = cxPix * 254 / cxMms / 10 ;
    yDpi = cyPix * 254 / cyMms / 10 ;

    DrawRuler (hdcEMF, 6 * xDpi, yDpi) ;

    hemf = CloseEnhMetaFile (hdcEMF) ;

    DeleteEnhMetaFile (hemf) ;
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
    ENHMETAHEADER   emh ;
    HENHMETAFILE    hemf ;
    int             cxImage, cyImage ;
    RECT            rect ;

    hemf = GetEnhMetaFile (TEXT ("emf8.emf")) ;

    GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;

    cxImage = emh.rclBounds.right - emh.rclBounds.left ;
    cyImage = emh.rclBounds.bottom - emh.rclBounds.top ;

    rect.left   = (cxArea - cxImage) / 2 ;
    rect.right  = (cxArea + cxImage) / 2 ;
```

```
    rect.top    = (cyArea - cyImage) / 2 ;
    rect.bottom = (cyArea + cyImage) / 2 ;

    PlayEnhMetaFile (hdc, hemf, &rect) ;

    DeleteEnhMetaFile (hemf) ;
}
```

EMF.C

```
/* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
    EMF.C——Enhanced Metafile Demonstration Shell Program
            (c) Charles Petzold, 1998
   - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

# include <windows.h>
# include <commdlg.h>
# include "..\\emf8\\resource.h"

extern void CreateRoutine (HWND) ;
extern void PaintRoutine (HWND, HDC, int, int) ;

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

HANDLE hInst ;

extern TCHAR szClass [] ;
extern TCHAR szTitle [] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    TCHAR       szResource [] = TEXT ("EMF") ;
    HWND        hwnd ;
    MSG         msg ;
    WNDCLASS    wndclass ;

    hInst = hInstance ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = szResource ;
    wndclass.lpszClassName  = szClass ;
```

```
    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szClass, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szClass, szTitle,
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;
    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

BOOL PrintRoutine (HWND hwnd)
{
    static DOCINFO     di ;
    static PRINTDLG    printdlg = { sizeof (PRINTDLG) } ;
    static TCHAR       szMessage [32] ;
    BOOL               bSuccess = FALSE ;
    HDC                hdcPrn ;
    int                cxPage, cyPage ;

    printdlg.Flags = PD_RETURNDC | PD_NOPAGENUMS | PD_NOSELECTION ;

    if (!PrintDlg (&printdlg))
        return TRUE ;

    if (NULL == (hdcPrn = printdlg.hDC))
        return FALSE ;

    cxPage = GetDeviceCaps (hdcPrn, HORZRES) ;
    cyPage = GetDeviceCaps (hdcPrn, VERTRES) ;

    lstrcpy (szMessage, szClass) ;
    lstrcat (szMessage, TEXT (": Printing")) ;

    di.cbSize       = sizeof (DOCINFO) ;
```

```
    di.lpszDocName = szMessage ;

    if (StartDoc (hdcPrn, &di) > 0)
    {
        if (StartPage (hdcPrn) > 0)
        {
            PaintRoutine (hwnd, hdcPrn, cxPage, cyPage) ;

            if (EndPage (hdcPrn) > 0)
            {
                EndDoc (hdcPrn) ;
                bSuccess = TRUE ;
            }
        }
    }
    DeleteDC (hdcPrn) ;

    return bSuccess ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    BOOL            bSuccess ;
    static int      cxClient, cyClient ;
    HDC             hdc ;
    PAINTSTRUCT     ps ;

    switch (message)
    {
    case WM_CREATE:
        CreateRoutine (hwnd) ;
        return 0 ;

    case WM_COMMAND:
        switch (wParam)
        {
        case IDM_PRINT:
            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
            ShowCursor (TRUE) ;

            bSuccess = PrintRoutine (hwnd) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

            if (!bSuccess)
                MessageBox (hwnd,
                            TEXT ("Error encountered during printing"),
```

```
                                szClass, MB_ICONASTERISK | MB_OK) ;
            return 0 ;

        case IDM_EXIT:
            SendMessage (hwnd, WM_CLOSE, 0, 0) ;
            return 0 ;

        case IDM_ABOUT:
            MessageBox (hwnd, TEXT ("Enhanced Metafile Demo Program \ n")
                              TEXT ("Copyright (c) Charles Petzold, 1998"),
                        szClass, MB_ICONINFORMATION | MB_OK) ;
            return 0 ;
        }
        break ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        PaintRoutine (hwnd, hdc, cxClient, cyClient) ;

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY :
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

EMF.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu

EMF MENU DISCARDABLE
BEGIN
    POPUP "&File"
```

```
    BEGIN
        MENUITEM "&Print...",                   IDM_PRINT
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       IDM_EXIT
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About...",                   IDM_ABOUT
    END
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by Emf.rc
//
#define IDM_PRINT                       40001
#define IDM_EXIT                        40002
#define IDM_ABOUT                       40003
```

在处理 WM_CREATE 消息期间,EMF.C 调用名为 CreateRoutine 的外部函数,该函数创建元文件。EMF.C 在两个地方调用 PaintRoutine 函数:一处在 WM_PAINT 消息期间,另一处在函数 PrintRoutine 中以响应菜单命令打印图像。

因为现代的打印机通常比视频显示器有更高的分辨率,打印元文件的能力是测试以特定大小提交图像能力的重要工具。当 EMF8 创建的元文件图像以特定大小显示时,最有意义。该图像是一把 6 英寸长 1 英寸宽的直尺,每英寸分为 16 格,数字从 1 到 5 为 TrueType 字体。

要绘制一把 6 英寸的直尺,需要知道一些设备分辨率的知识。EMF8.C 中的 CreateRoutine 函数首先创建元文件,然后使用从 CreateEnhMetaFile 返回的设备描述表句柄调用 GetDeviceCaps 4 次。这些调用获取单位分别为 mm 和像素的显示平面的高度与宽度。

这听起来也许有些奇怪。元文件设备描述表通常被看作是 GDI 绘图命令的存储媒介,它不是像视频显示器或打印机那样的真正的设备,因此怎样才能获得它的宽度和高度呢?

您可能已经想起来了,CreateEnhMetaFile 的第一个参数被称作"参考设备描述表"。GDI 用这为元文件建立设备特征。如果参数设置为 NULL(如 EMF8 中),GDI 就把显示器作为参考设备描述表。因而,当 EMF8 使用设备描述表调用 GetDeviceCaps 时,它实际上获取有关显示器的信息。

EMF8.C 以像素大小除以毫米大小并乘以 25.4(1 英寸为 25.4 mm)来计算以每英寸的点数为单位的分辨率。

即使我们非常认真地以元文件直尺的正确大小绘制它，但还是没有完成任务。在提交图像的时候，PlayEnhMetaFile 函数在显示图像时，使用作为最后一个参数传递给它的矩形伸展图像，因此该矩形必须设置为直尺的大小。

由于这个原因，EMF8 中的 PaintRoutine 函数调用 GetEnhMetaFileHeader 函数来获取元文件的头信息。ENHMETAHEADER 结构的 rclBounds 字段指出以像素为单位的元文件图像的边框矩形的大小。该程序使用这一信息使直尺位于客户区中央，如图 18-6 所示。

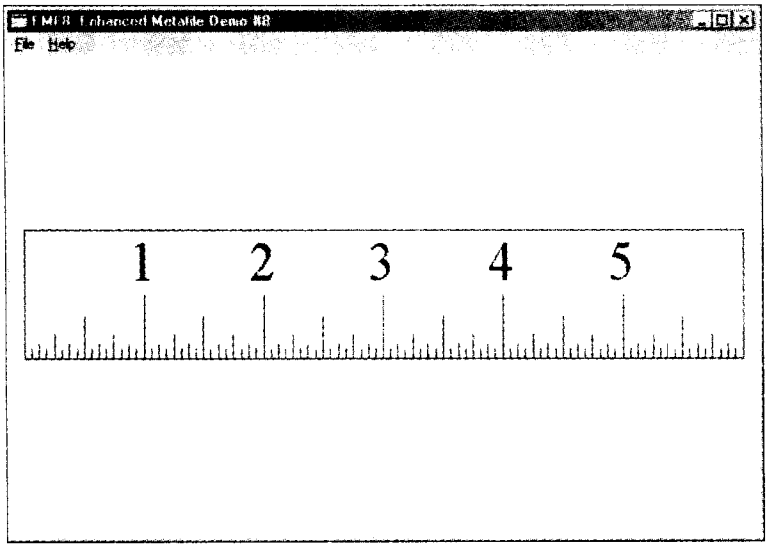


图 18-6　EMF8 显示

记住，如果您拿直尺与屏幕中的直尺比较时，两者并不一定非常吻合。如同第五章中所论述的，显示器只能近似地显示实际尺寸。

这个技术好像已经起作用了，现在就来试着打印图像。哇，如果您有一台 300 dpi 的激光打印机，那么打印出的直尺的宽为 $1\frac{1}{3}$ 英寸。这是由于我们使用了基于视频显示器的像素尺寸。虽然您可能认为这把小尺子很可爱，但它不是我们所需要的。让我们再试一试。

ENHMETAHEADER 结构包括两个描述图像大小的矩形结构。第一个是 rclBounds，EMF8 使用这个，它以像素为单位给出图像的大小。第二为 rclFrame，它以 0.01 mm 为单位给出图像的大小。这两个字段之间的关系是由最初创建元文件时使用的参考设备描述表决定，在此情况下为显示器。（元文件头也包括两个名为 szlDevice 和 szlMillimeters 的字段，它们是 SIZEL 结构，分别以像素单位和 mm 单位指出了参考设备的大小，这与

从 GetDeviceCaps 得到的信息一样。)

EMF9 使用的是关于图像的毫米大小的信息,如程序 18-11 所示。

程序 18-11　EMF9 程序

EMF9.C

```
/*-------------------
   EMF9.C——Enhanced Metafile Demo #9
            (c) Charles Petzold, 1998
   -------------------*/

#include <windows.h>
#include <string.h>

TCHAR szClass[] = TEXT("EMF9");
TCHAR szTitle[] = TEXT("EMF9: Enhanced Metafile Demo #9");

void CreateRoutine(HWND hwnd)
{
}

void PaintRoutine(HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     ENHMETAHEADER  emh;
     HENHMETAFILE   hemf;
     int            cxMms, cyMms, cxPix, cyPix, cxImage, cyImage;
     RECT           rect;

     cxMms = GetDeviceCaps(hdc, HORZSIZE);
     cyMms = GetDeviceCaps(hdc, VERTSIZE);
     cxPix = GetDeviceCaps(hdc, HORZRES);
     cyPix = GetDeviceCaps(hdc, VERTRES);

     hemf = GetEnhMetaFile(TEXT("..\\emf8\\emf8.emf"));

     GetEnhMetaFileHeader(hemf, sizeof(emh), &emh);

     cxImage = emh.rclFrame.right - emh.rclFrame.left;
     cyImage = emh.rclFrame.bottom - emh.rclFrame.top;

     cxImage = cxImage * cxPix / cxMms / 100;
     cyImage = cyImage * cyPix / cyMms / 100;

     rect.left   = (cxArea - cxImage) / 2;
     rect.right  = (cxArea + cxImage) / 2;
     rect.top    = (cyArea - cyImage) / 2;
     rect.bottom = (cyArea + cyImage) / 2;
```

```
    PlayEnhMetaFile (hdc, hemf, &rect) ;

    DeleteEnhMetaFile (hemf) ;
}
```

EMF9 使用由 EMF8 创建的元文件，因此确定运行了 EMF8。

EMF9 中的 PaintRoutine 函数首先使用目的设备描述表调用 GetDeviceCaps 4 次。像在 EMF8 中的 CreateRoutine 函数一样，这些调用提供有关设备分辨率的信息。在得到元文件句柄之后，它获取头结构并使用 rclFrame 字段来计算以 0.01 mm 为单位的元文件图像大小。这是第一步。

然后，函数通过乘以输出设备的像素大小、除以毫米大小，再除以 100(因为度量尺寸以 0.01 mm 为单位)，将此大小转换为像素大小。现在，PaintRoutine 函数具有以像素为单位的直尺大小——与显示器无关。这是适合目的设备的像素大小，而且很容易使图像居中。

就显示器而言，EMF9 的显示与 EMF8 显示的一样。但是如果从 EMF9 打印直尺，您会看到更正常的直尺——6 英寸长，1 英寸宽。

18.2.9　缩放比和纵横比

您也可能想要使用 EMF8 创建的直尺元文件，而不必显示 6 英寸的图像。保持图像正确的 6:1 的纵横比是重要的。如前所述，在字处理程序或别的应用程序中使用围绕框来改变元文件的大小是很方便的，但是这样会导致某种失真。在这种应用程序中，应该给用户一个选项来保持原先的纵横比，而无论围绕框的大小如何变化。这就是说，传递给 PlayEnhMetaFile 的矩形结构不能直接由用户选择的围绕框定义。传递给该函数的矩形结构只是围绕框的一部分。

让我们看一看程序 18-12 所示的 EMF10 是如何做的。

程序 18-12　EMF10 程序

EMF10.C

```
/*-------------------------
   EMF10.C——Enhanced Metafile Demo #10
            (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>

TCHAR szClass [] = TEXT ("EMF10") ;
TCHAR szTitle [] = TEXT ("EMF10: Enhanced Metafile Demo #10") ;
```

```
void CreateRoutine (HWND hwnd)
{
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     ENHMETAHEADER  emh ;
     float          fScale ;
     HENHMETAFILE   hemf ;
     int            cxMms, cyMms, cxPix, cyPix, cxImage, cyImage ;
     RECT           rect ;

     cxMms = GetDeviceCaps (hdc, HORZSIZE) ;
     cyMms = GetDeviceCaps (hdc, VERTSIZE) ;
     cxPix = GetDeviceCaps (hdc, HORZRES) ;
     cyPix = GetDeviceCaps (hdc, VERTRES) ;

     hemf = GetEnhMetaFile (TEXT ("..\\emf8\\emf8.emf")) ;

     GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;

     cxImage = emh.rclFrame.right - emh.rclFrame.left ;
     cyImage = emh.rclFrame.bottom - emh.rclFrame.top ;

     cxImage = cxImage * cxPix / cxMms / 100 ;
     cyImage = cyImage * cyPix / cyMms / 100 ;

     fScale = min ((float) cxArea / cxImage, (float) cyArea / cyImage) ;

     cxImage = (int) (fScale * cxImage) ;
     cyImage = (int) (fScale * cyImage) ;

     rect.left   = (cxArea - cxImage) / 2 ;
     rect.right  = (cxArea + cxImage) / 2 ;
     rect.top    = (cyArea - cyImage) / 2 ;
     rect.bottom = (cyArea + cyImage) / 2 ;

     PlayEnhMetaFile (hdc, hemf, &rect) ;

     DeleteEnhMetaFile (hemf) ;
}
```

EMF10伸展直尺图像以适应客户区(或打印页面的可打印部分),但不会失真。通常直尺会伸展到客户区的整个宽度,但是会上下居中。如果使窗口太矮,则直尺会与客户区一般高,但是会水平居中。

可能有多种方法来计算合适的显示矩形,但是我们在EMF9的基础上完成该项工作。EMF10.C中的PaintRoutine函数开始部分与EMF9.C相同,为目的设备描述表计算

6 英寸长的直尺图像适当的像素大小。

然后，程序计算名为 fScale 的浮点值，它是客户区宽度与图像宽度的比值以及客户区高度与图像高度比值两者的最小值。这个因子在计算围绕矩形前用于增加图像的像素大小。

18.2.10　元文件中的映射方式

前面绘制的直尺单位有英寸，也有 mm。这种工作使用 GDI 提供的各种映射方式似乎非常适合。但是我坚持使用像素，并“手工”完成所有必要的计算。为什么呢?

答案很简单，就是将映射方式与元文件一起使用会十分混乱。我们不妨试用一下。

当使用元文件设备描述表调用 SetMapMode 时，该函数在元文件中像其他 GDI 函数一样被编码。如程序 18-13 所示的 EMF11 显示的那样。

程序 18-13　EMF11 程序

EMF11.C

```
/*-----------------------
   EMF11.C——Enhanced Metafile Demo #11
            (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

TCHAR szClass[] = TEXT("EMF11");
TCHAR szTitle[] = TEXT("EMF11: Enhanced Metafile Demo #11");

void DrawRuler(HDC hdc, int cx, int cy)
{
     int       i, iHeight;
     LOGFONT   lf;
     TCHAR     ch;

          // Black pen with 1-point width

     SelectObject(hdc, CreatePen(PS_SOLID, cx / 72 / 6, 0));
          // Rectangle surrounding entire pen (with adjustment)

     if (GetVersion() & 0x80000000)                    // Windows 98
          Rectangle(hdc, 0, -2, cx + 2, cy);
     else                                              // Windows NT
          Rectangle(hdc, 0, -1, cx + 1, cy);

          // Tick marks
```

```
    for (i = 1 ; i < 96 ; i++)
    {
          if (i % 16 == 0) iHeight = cy /  2 ;     // inches
        else if (i %  8 == 0) iHeight = cy /  3 ;     // half inches
        else if (i %  4 == 0) iHeight = cy /  5 ;     // quarter inches
        else if (i %  2 == 0) iHeight = cy /  8 ;     // eighths
        else                  iHeight = cy / 12 ;     // sixteenths
        MoveToEx (hdc, i * cx / 96, 0, NULL) ;
        LineTo   (hdc, i * cx / 96, iHeight) ;
    }
        // Create logical font

    FillMemory (&lf, sizeof (lf), 0) ;
    lf.lfHeight = cy / 2 ;
    lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;

    SelectObject (hdc, CreateFontIndirect (&lf)) ;
    SetTextAlign (hdc, TA_BOTTOM | TA_CENTER) ;
    SetBkMode (hdc, TRANSPARENT) ;

        // Display numbers

    for (i = 1 ; i <= 5 ; i++)
    {
        ch = (TCHAR) (i + '0') ;
        TextOut (hdc, i * cx / 6, cy / 2, &ch, 1) ;
    }
        // Clean up

    DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
    DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN))) ;
}

void CreateRoutine (HWND hwnd)
{
    HDC            hdcEMF ;
    HENHMETAFILE   hemf ;

    hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf11.emf"), NULL,
                                TEXT ("EMF11 \0EMF Demo # 11 \0")) ;

    SetMapMode (hdcEMF, MM_LOENGLISH) ;

    DrawRuler (hdcEMF, 600, 100) ;

    hemf = CloseEnhMetaFile (hdcEMF) ;

    DeleteEnhMetaFile (hemf) ;
```

```
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     ENHMETAHEADER  emh ;
     HENHMETAFILE   hemf ;
     int            cxMms, cyMms, cxPix, cyPix, cxImage, cyImage ;
     RECT           rect ;

     cxMms = GetDeviceCaps (hdc, HORZSIZE) ;
     cyMms = GetDeviceCaps (hdc, VERTSIZE) ;
     cxPix = GetDeviceCaps (hdc, HORZRES) ;
     cyPix = GetDeviceCaps (hdc, VERTRES) ;

     hemf = GetEnhMetaFile (TEXT ("emf11.emf")) ;

     GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;

     cxImage = emh.rclFrame.right - emh.rclFrame.left ;
     cyImage = emh.rclFrame.bottom - emh.rclFrame.top ;

     cxImage = cxImage * cxPix / cxMms / 100 ;
     cyImage = cyImage * cyPix / cyMms / 100 ;

     rect.left   = (cxArea - cxImage) / 2 ;
     rect.top    = (cyArea - cyImage) / 2 ;
     rect.right  = (cxArea + cxImage) / 2 ;
     rect.bottom = (cyArea + cyImage) / 2 ;

     PlayEnhMetaFile (hdc, hemf, &rect) ;

     DeleteEnhMetaFile (hemf) ;
}
```

EMF11 中的 CreateRoutine 函数比 EMF8(最初的直尺元文件程序)中的那个简单,因为它不需要调用 GetDeviceCaps 来确定以每英寸点数为单位的显示器分辨率。相反,EMF11 调用 SetMapMode 将映射方式设置为 MM_LOENGLISH,其逻辑单位等于0.01英寸。因而,直尺的大小为 600 × 100 个单位,并将这些数值传递给 DrawRuler。

除了 MoveToEx 和 LineTo 调用绘制直尺的刻度外,EMF11 中的 DrawRuler 函数与 EMF9 中的一样。当以像素单位绘制时(默认的 MM_TEXT 映射方式),垂直轴上的单位沿着屏幕向下增长。对于 MM_LOENGLISH 映射方式(以及其他度量映射方式),则向上增长。这就需要修改程序代码。同时,也需要修改 Rectangle 函数中的调节因子。

EMF11 中的 PaintRoutine 函数基本上与 EMF9 中的相同,那个版本的程序能在显示器和打印机上以正确尺寸显示直尺。唯一不同之处在于 EMF11 使用 EMF11.EMF 文

件,而 EMF9 使用 EMF8 创建的 EMF8.EMF 文件。

EMF11 显示的图像基本上与 EMF9 所显示的相同。因此,在这里可以看到将 SetMapMode 调用嵌入元文件能够简化元文件的创建,而且不影响以其正确大小显示元文件的机制。

18.2.11　映射和显示

在 EMF11 中计算目的矩形包括对 GetDeviceCaps 的几个调用。我们的第二个目标是使用映射方式代替这些调用。GDI 将目的矩形的坐标视为逻辑坐标。为这些坐标使用 MM_HIMETRIC 似乎是个好方案,因为它使用 0.01 mm 作为逻辑单位,与增强型元文件头中用于围绕矩形的单位相同。

程序 18-14 中所示的 EMF12 程序,保留了 EMF8 中使用的 DrawRuler 逻辑,但是使用 MM_HIMETRIC 映射方式显示元文件。

程序 18-14　EMF12 程序

EMF12.C

```
/*-----------------------
    EMF12.C——Enhanced Metafile Demo #12
              (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

TCHAR szClass[] = TEXT("EMF12");
TCHAR szTitle[] = TEXT("EMF12: Enhanced Metafile Demo #12");

void DrawRuler(HDC hdc, int cx, int cy)
{
     int        iAdj, i, iHeight;
     LOGFONT    lf;
     TCHAR      ch;

     iAdj = GetVersion() & 0x80000000 ? 0 : 1;

          // Black pen with 1-point width

     SelectObject(hdc, CreatePen(PS_SOLID, cx / 72 / 6, 0));

          // Rectangle surrounding entire pen (with adjustment)

     Rectangle(hdc, iAdj, iAdj, cx + iAdj + 1, cy + iAdj + 1);
```

```
        // Tick marks

    for (i = 1 ; i < 96 ; i++)
    {
            if (i % 16 == 0) iHeight = cy /  2 ;     // inches
        else if (i %  8 == 0) iHeight = cy /  3 ;     // half inches
        else if (i %  4 == 0) iHeight = cy /  5 ;     // quarter inches
        else if (i %  2 == 0) iHeight = cy /  8 ;     // eighths
        else                  iHeight = cy / 12 ;     // sixteenths

        MoveToEx (hdc, i * cx / 96, cy, NULL) ;
        LineTo   (hdc, i * cx / 96, cy - iHeight) ;
    }
        // Create logical font

    FillMemory (&lf, sizeof (lf), 0) ;
    lf.lfHeight = cy / 2 ;
    lstrcpy (lf.lfFaceName, TEXT ("Times New Roman")) ;

    SelectObject (hdc, CreateFontIndirect (&lf)) ;
    SetTextAlign (hdc, TA_BOTTOM | TA_CENTER) ;
    SetBkMode (hdc, TRANSPARENT) ;

        // Display numbers

    for (i = 1 ; i <= 5 ; i++)
    {
        ch = (TCHAR) (i + '0') ;
        TextOut (hdc, i * cx / 6, cy / 2, &ch, 1) ;
    }
        // Clean up

    DeleteObject (SelectObject (hdc, GetStockObject (SYSTEM_FONT))) ;
    DeleteObject (SelectObject (hdc, GetStockObject (BLACK_PEN))) ;
}

void CreateRoutine (HWND hwnd)
{
    HDC             hdcEMF ;
    HENHMETAFILE    hemf ;
    int             cxMms, cyMms, cxPix, cyPix, xDpi, yDpi ;

    hdcEMF = CreateEnhMetaFile (NULL, TEXT ("emf12.emf"), NULL,
                                TEXT ("EMF13 \ 0EMF Demo # 12 \ 0")) ;

    cxMms = GetDeviceCaps (hdcEMF, HORZSIZE) ;
    cyMms = GetDeviceCaps (hdcEMF, VERTSIZE) ;
    cxPix = GetDeviceCaps (hdcEMF, HORZRES) ;
```

```
    cyPix = GetDeviceCaps (hdcEMF, VERTRES) ;

    xDpi = cxPix * 254 / cxMms / 10 ;
    yDpi = cyPix * 254 / cyMms / 10 ;

    DrawRuler (hdcEMF, 6 * xDpi, yDpi) ;

    hemf = CloseEnhMetaFile (hdcEMF) ;

    DeleteEnhMetaFile (hemf) ;
}

void PaintRoutine (HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
    ENHMETAHEADER  emh ;
    HENHMETAFILE   hemf ;
    POINT          pt ;
    int            cxImage, cyImage ;
    RECT           rect ;

    SetMapMode (hdc, MM_HIMETRIC) ;

    SetViewportOrgEx (hdc, 0, cyArea, NULL) ;

    pt.x = cxArea ;
    pt.y = 0 ;

    DPtoLP (hdc, &pt, 1) ;

    hemf = GetEnhMetaFile (TEXT ("emf12.emf")) ;

    GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;

    cxImage = emh.rclFrame.right - emh.rclFrame.left ;
    cyImage = emh.rclFrame.bottom - emh.rclFrame.top ;

    rect.left   = (pt.x - cxImage) / 2 ;
    rect.top    = (pt.y + cyImage) / 2 ;
    rect.right  = (pt.x + cxImage) / 2 ;
    rect.bottom = (pt.y - cyImage) / 2 ;

    PlayEnhMetaFile (hdc, hemf, &rect) ;

    DeleteEnhMetaFile (hemf) ;
}
```

EMF12 中的 PaintRoutine 函数首先将映射方式设置为 MM_HIMETRIC。像其他度量映射方式一样，y 值沿着屏幕向上增长。然而，原点坐标仍在屏幕的左上角，这就意味

客户区内的 y 坐标值是负数。为了纠正这个问题，程序调用 SetViewportOrgEx 将原点坐标设置在左下角。

设备坐标(cxArea,0)位于屏幕的右上角。把该坐标点传递给 DPtoLP("设备坐标点到逻辑坐标点")函数，得到以 0.01 mm 为单位的客户区大小。

然后，程序加载元文件，取得文件头，并找到以 0.01 mm 为单位的元文件大小。这样计算目的矩形在客户区居中的位置就变得十分简单。

现在我们看到了在创建元文件时能够使用映射方式，显示它时也能使用映射方式。我们能一起完成它们吗？

如程序 18-15 所示的 EMF13 演示的那样，这是可以的。

程序 18-15　EMF13 程序

EMF13.C

```
/*-----------------------
   EMF13.C——Enhanced Metafile Demo #13
            (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

TCHAR szClass[] = TEXT("EMF13");
TCHAR szTitle[] = TEXT("EMF13: Enhanced Metafile Demo #13");

void CreateRoutine(HWND hwnd)
{
}

void PaintRoutine(HWND hwnd, HDC hdc, int cxArea, int cyArea)
{
     ENHMETAHEADER  emh;
     HENHMETAFILE   hemf;
     POINT          pt;
     int            cxImage, cyImage;
     RECT           rect;

     SetMapMode(hdc, MM_HIMETRIC);

     SetViewportOrgEx(hdc, 0, cyArea, NULL);

     pt.x = cxArea;
     pt.y = 0;

     DPtoLP(hdc, &pt, 1);
```

```
    hemf = GetEnhMetaFile (TEXT ("..\\emf11\\emf11.emf")) ;

    GetEnhMetaFileHeader (hemf, sizeof (emh), &emh) ;

    cxImage = emh.rclFrame.right - emh.rclFrame.left ;
    cyImage = emh.rclFrame.bottom - emh.rclFrame.top ;

    rect.left   = (pt.x - cxImage) / 2 ;
    rect.top    = (pt.y + cyImage) / 2 ;
    rect.right  = (pt.x + cxImage) / 2 ;
    rect.bottom = (pt.y - cyImage) / 2 ;

    PlayEnhMetaFile (hdc, hemf, &rect) ;

    DeleteEnhMetaFile (hemf) ;
}
```

在 EMF13 中，由于直尺元文件已由 EMF11 创建，所以它没有使用映射方式创建元文件。EMF13 只是简单地加载元文件，然后像 EMF11 一样使用映射方式计算目的矩形。

现在，我们可以建立一些规则。在创建元文件时，GDI 使用对映射方式的任意嵌入的更改，来计算以像素和 mm 为单位的元文件图像的大小。图像的大小存储在元文件头内。在显示元文件时，GDI 在调用 PlayEnhMetaFile 时根据有效的映射方式建立目的矩形的物理位置。元文件中的任何记录也无法更改该位置。

第三部分　高级主题

第十九章

多文档界面

多文档界面(MDI)是 Microsoft Windows 文档处理应用程序的一种规范,该规范描述了窗口结构和允许用户在单个应用程序中使用多个文档的用户界面(如字处理程序中的文本文档和电子表格程序中的电子表格)。简单地说,就像 Windows 在一个屏幕上维护多个应用程序窗口一样,MDI 应用程序在一个客户区内维护多个文档窗口。Windows 中的第一个 MDI 应用程序是 Windows 下的 Microsoft Excel 的第一个版本,紧接着又出现了许多其他的应用程序。

19.1　MDI 概念

尽管 MDI 规范随着 Windows 2.0 的推出已经很普及,但在那时,MDI 应用程序写起来很困难,并且需要一些非常复杂的程序设计工作。从 Windows 3.0 起,其中许多工作就都已经为您做好了。Windows 95 中增强的支持也已经被添加进 Windows 98 和 Microsoft Windows NT 中。

19.1.1　MDI 的组成

MDI 程序的主应用程序窗口是很相似的:有一个标题栏、一个菜单、一个缩放边框、一个系统菜单图标,以及最大化/最小化/关闭按钮。客户区经常被称为"工作空间",它不直接用于显示程序输出。这个工作空间包括零个或多个子窗口,每个窗口都显示一个文档。

这些子窗口看起来与通常的应用程序窗口及 MDI 程序的主窗口很相似。它们有一个标题栏、一个缩放边框、一个系统菜单图标,以及最大化/最小化/关闭按钮,可能还包括滚动条。但是文档窗口没有菜单,主应用程序窗口上的菜单适用于文档窗口。

在任何时候都只能有一个文档窗口是活动的(加亮标题栏来表示),它出现在其他所有文档窗口之前。所有文档窗口都由工作空间区域加以剪裁,而不会出现在应用程序窗口之外。

对 Windows 程序员来说,初看起来 MDI 似乎相当简单。需要程序员做的工作好像就是为每个文档创建一个 WS_CHILD 窗口,并使程序的主应用程序窗口成为文档窗口的父窗口。但对现有的 MDI 应用程序稍加研究,就会发现一些导致编码困难的复杂问题,例如:

- MDI 文档窗口可以被最小化。它的图标出现在工作空间的底部的短标题栏中。一般来说,MDI 应用程序可以将不同的图标分别用于主应用程序窗口和每一类文档应用。
- MDI 文档窗口可以被最大化。在这种情况下,文档窗口的标题栏(一般用来显示窗口中文档的文件名)消失,文件名出现在应用程序窗口标题栏的应用程序名之后,文档窗口的系统菜单图标成为应用程序窗口的顶层菜单中的第一项。关闭文档窗口按钮变成顶层菜单中的最后一项,且出现在最右边。
- 用以关闭文档窗口的系统键盘加速键与关闭主窗口的系统键盘加速键一样,只是 Ctrl 键代替了 Alt 键。这也就是说,Alt + F4 用于关闭应用程序窗口,而 Ctrl + F4 用于关闭文档窗口。此外,Ctrl + F6 可以在活动 MDI 应用程序的子文档窗口之间切换。与平时一样,Alt + 空格键激活主窗口的系统菜单,Alt + -(减号)激活活动子文档窗口的系统菜单。
- 当使用光标键在菜单项间移动时,控制通常从系统菜单转到菜单栏中的第一项。在 MDI 应用程序中,控制是从应用程序系统菜单转到活动文档系统菜单,然后再转到菜单栏中的第一项。
- 如果应用程序能够支持若干种类型的子窗口(如 Microsoft Excel 中的工作表和图表文档),那么菜单应能反映出与这种类型的文档有关的操作。这就要求当不同的文本窗口变成活动窗口时,程序能更换菜单。此外,当没有文档窗口存在时,菜单应该被缩减到只剩下与打开新文档有关的操作。
- 顶层菜单上有一个被叫做“窗口(Window)”的菜单项。按照习惯,这是顶层菜单上“帮助(Help)”之前的那一项,即倒数第二项。“窗口”子菜单上通常包含一些在工作空间内安排文档窗口的选项。文档窗口可以从左上方开始“层叠”,也可以并排“平铺”以便使每一个文档窗口都是完全可见的。这个子菜单同时也包含一个所有文档窗口的列表。从中选择一个文档窗口,就可以把此文档窗口移到前台,使之成为活动窗口。

Windows 98 支持 MDI 的所有这些方面。当然,需要您做一些工作(如下面的示例程序所示),但是,这绝不是要您编写大量代码来直接支持所有这些功能。

19.1.2 MDI 支持

探讨 Windows 的 MDI 支持时需要引入一些新术语。主应用程序窗口称为“框架窗口”，就像传统的 Windows 程序一样，它是 WS_OVERLAPPEDWINDOW 风格的窗口。

MDI 应用程序还根据预定义的窗口类 MDICLIENT 创建“客户窗口”，这一客户窗口是用这种窗口类和 WS_CHILD 风格调用 CreateWindow 来创建的。这一调用的最后一个参数是指向一个 CLIENTCREATESTRUCT 类型的结构的指针。这个客户窗口覆盖框架窗口的客户区，并提供许多 MDI 支持。此客户窗口的颜色是系统颜色 COLOR_APPWORKSPACE。

文档窗口被称为“子窗口”。通过初始化一个 MDICREATESTRUCT 类型的结构，以一个指向此结构的指针为参数将消息 WM_MDICREATE 发送给客户窗口，就可以创建这些文档窗口。

文档窗口是客户窗口的子窗口，而客户窗口又是框架窗口的子窗口。父-子窗口层次结构如图 19-1 所示。

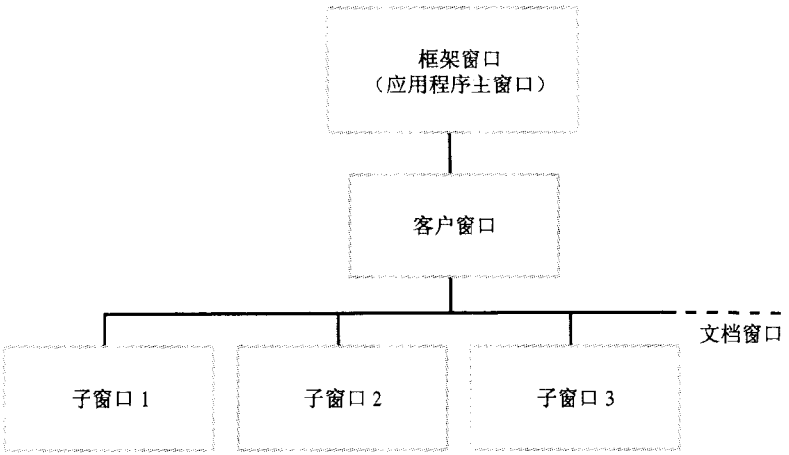


图 19-1　Windows MDI 应用程序的父-子层次结构图

您需要框架窗口的窗口类(及窗口过程)和一个由应用程序支持的每类子窗口的窗口类(及窗口过程)。由于已经预先注册了窗口类，所以不需要客户窗口的窗口过程。

Windows 98 的 MDI 支持包括 1 个窗口类、5 个函数、两个数据结构和 12 个消息。前面已经提到了 MDI 窗口类，即 MDICLIENT，以及数据结构 CLIENTCREATESTRUCT 和 MDICREATESTRUCT。在 MDI 应用程序中，这 5 个函数中的两个用于取代 DefWindowProc：不再将 DefWindowProc 调用用于所有未处理的消息，而是由框架窗口过程调用 Def-

FrameProc,子窗口过程调用 DefMDIChildProc。另一个 MDI 特有的函数 TranslateMDISysAccel 与第十章中讨论的 TranslateAccelerator 的使用方式相同。MDI 支持也包括 ArrangeIconicWindows 函数,但有一个专用的 MDI 消息使得此函数对 MDI 程序来说不再必要。

第 5 个 MDI 函数是 CreateMDIWindow,它使得子窗口可以在单独的执行线程中被创建。在单线程的程序中不需要这个函数,我会演示这一点。

在下面的程序中,我将演示 12 个 MDI 消息中的 9 个(其他 3 个消息一般不用),这些消息的前缀是 WM_MDI。框架窗口向客户窗口发送其中某个消息,以便在子窗口上完成一项操作或者获取关于子窗口的信息(例如,框架窗口发送一个 WM_MDICREATE 消息给客户窗口,以创建子窗口)。消息 WM_MDIACTIVATE 是一特例:框架窗口可以将该消息发送给客户窗口以激活一个子窗口,而客户窗口也把该消息发送给将被激活或被取消激活的子窗口,以便通知它们这一变化。

19.2 MDI 的示例程序

程序 19-1 所示的 MDIDEMO 程序说明了编写 MDI 应用程序的基本方法。

程序 19-1　MDIDEMO 程序

MDIDEMO.C

```
/*-------------------------
   MDIDEMO.C——Multiple-Document Interface Demonstration
              (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include "resource.h"

#define INIT_MENU_POS    0
#define HELLO_MENU_POS   2
#define RECT_MENU_POS    1

#define IDM_FIRSTCHILD 50000

LRESULT  CALLBACK FrameWndProc   (HWND, UINT, WPARAM, LPARAM) ;
BOOL     CALLBACK CloseEnumProc  (HWND, LPARAM) ;
LRESULT  CALLBACK HelloWndProc   (HWND, UINT, WPARAM, LPARAM) ;
LRESULT  CALLBACK RectWndProc    (HWND, UINT, WPARAM, LPARAM) ;

    // structure for storing data unique to each Hello child window
```

```
typedef struct tagHELLODATA
{
     UINT       iColor ;
     COLORREF   clrText ;
}
HELLODATA, * PHELLODATA ;

    // structure for storing data unique to each Rect child window

typedef struct tagRECTDATA
{
     short cxClient ;
     short cyClient ;
}
RECTDATA, * PRECTDATA ;
    // global variables

TCHAR      szAppName[ ]   = TEXT ("MDIDemo") ;
TCHAR      szFrameClass[ ] = TEXT ("MdiFrame") ;
TCHAR      szHelloClass[ ]  = TEXT ("MdiHelloChild") ;
TCHAR      szRectClass[ ]   = TEXT ("MdiRectChild") ;
HINSTANCE  hInst ;
HMENU      hMenuInit, hMenuHello, hMenuRect ;
HMENU      hMenuInitWindow, hMenuHelloWindow, hMenuRectWindow ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HACCEL      hAccel ;
     HWND        hwndFrame, hwndClient ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     hInst = hInstance ;

         // Register the frame window class

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = FrameWndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) (COLOR_APPWORKSPACE + 1) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szFrameClass ;
```

```
if (!RegisterClass (&wndclass))
{
     MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                 szAppName, MB_ICONERROR) ;
     return 0 ;
}

    // Register the Hello child window class

wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc   = HelloWndProc ;
wndclass.cbClsExtra    = 0 ;
wndclass.cbWndExtra    = sizeof (HANDLE) ;
wndclass.hInstance     = hInstance ;
wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
wndclass.lpszMenuName  = NULL ;
wndclass.lpszClassName = szHelloClass ;

RegisterClass (&wndclass) ;

    // Register the Rect child window class

wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
wndclass.lpfnWndProc   = RectWndProc ;
wndclass.cbClsExtra    = 0 ;
wndclass.cbWndExtra    = sizeof (HANDLE) ;
wndclass.hInstance     = hInstance ;
wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
wndclass.lpszMenuName  = NULL ;
wndclass.lpszClassName = szRectClass ;

RegisterClass (&wndclass) ;

    // Obtain handles to three possible menus & submenus

hMenuInit  = LoadMenu (hInstance, TEXT ("MdiMenuInit")) ;
hMenuHello = LoadMenu (hInstance, TEXT ("MdiMenuHello")) ;
hMenuRect  = LoadMenu (hInstance, TEXT ("MdiMenuRect")) ;

hMenuInitWindow  = GetSubMenu (hMenuInit,  INIT_MENU_POS) ;
hMenuHelloWindow = GetSubMenu (hMenuHello, HELLO_MENU_POS) ;
hMenuRectWindow  = GetSubMenu (hMenuRect,  RECT_MENU_POS) ;
```

```
        // Load accelerator table

    hAccel = LoadAccelerators (hInstance, szAppName) ;

        // Create the frame window

    hwndFrame = CreateWindow (szFrameClass, TEXT ("MDI Demonstration"),
                              WS_OVERLAPPEDWINDOW | WS_CLIPCHILDREN,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              CW_USEDEFAULT, CW_USEDEFAULT,
                              NULL, hMenuInit, hInstance, NULL) ;
    hwndClient = GetWindow (hwndFrame, GW_CHILD) ;

    ShowWindow (hwndFrame, iCmdShow) ;
    UpdateWindow (hwndFrame) ;

        // Enter the modified message loop

    while (GetMessage (&msg, NULL, 0, 0))
    {
        if (!TranslateMDISysAccel (hwndClient, &msg) &&
            !TranslateAccelerator (hwndFrame, hAccel, &msg))
        {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
    }
        // Clean up by deleting unattached menus

    DestroyMenu (hMenuHello) ;
    DestroyMenu (hMenuRect) ;

    return msg.wParam ;
    }

LRESULT CALLBACK FrameWndProc (HWND hwnd, UINT message,
                               WPARAM wParam, LPARAM lParam)
{
    static HWND             hwndClient ;
    CLIENTCREATESTRUCT      clientcreate ;
    HWND                    hwndChild ;
    MDICREATESTRUCT         mdicreate ;

    switch (message)
    {
    case WM_CREATE:                 // Create the client window

        clientcreate.hWindowMenu  = hMenuInitWindow ;
```

```
        clientcreate.idFirstChild = IDM_FIRSTCHILD ;

        hwndClient = CreateWindow (TEXT ("MDICLIENT"), NULL,
                                   WS_CHILD | WS_CLIPCHILDREN | WS_VISIBLE,
                                   0, 0, 0, 0, hwnd, (HMENU) 1, hInst,
                                   (PSTR) &clientcreate) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_NEWHELLO:         // Create a Hello child window

            mdicreate.szClass = szHelloClass ;
            mdicreate.szTitle  = TEXT ("Hello") ;
            mdicreate.hOwner = hInst ;
            mdicreate.x       = CW_USEDEFAULT ;
            mdicreate.y       = CW_USEDEFAULT ;
            mdicreate.cx      = CW_USEDEFAULT ;
            mdicreate.cy      = CW_USEDEFAULT ;
            mdicreate.style   = 0 ;
            mdicreate.lParam  = 0 ;

            hwndChild = (HWND) SendMessage (hwndClient,
                                WM_MDICREATE, 0,
                                (LPARAM) (LPMDICREATESTRUCT) &mdicreate) ;
            return 0 ;

        case IDM_FILE_NEWRECT:          // Create a Rect child window

            mdicreate.szClass = szRectClass ;
            mdicreate.szTitle  = TEXT ("Rectangles") ;
            mdicreate.hOwner = hInst ;
            mdicreate.x       = CW_USEDEFAULT ;
            mdicreate.y       = CW_USEDEFAULT ;
            mdicreate.cx      = CW_USEDEFAULT ;
            mdicreate.cy      = CW_USEDEFAULT ;
            mdicreate.style   = 0 ;
            mdicreate.lParam  = 0 ;

            hwndChild = (HWND) SendMessage (hwndClient,
                                WM_MDICREATE, 0,
                                (LPARAM) (LPMDICREATESTRUCT) &mdicreate) ;
            return 0 ;

        case IDM_FILE_CLOSE:            // Close the active window
```

```
            hwndChild = (HWND) SendMessage (hwndClient,
                                            WM_MDIGETACTIVE, 0, 0) ;

            if (SendMessage (hwndChild, WM_QUERYENDSESSION, 0, 0))
                SendMessage (hwndClient, WM_MDIDESTROY,
                             (WPARAM) hwndChild, 0) ;
            return 0 ;

        case IDM_APP_EXIT:          // Exit the program

            SendMessage (hwnd, WM_CLOSE, 0, 0) ;
            return 0 ;

            // messages for arranging windows

        case IDM_WINDOW_TILE:
            SendMessage (hwndClient, WM_MDITILE, 0, 0) ;
            return 0 ;

        case IDM_WINDOW_CASCADE:
            SendMessage (hwndClient, WM_MDICASCADE, 0, 0) ;
            return 0 ;

        case IDM_WINDOW_ARRANGE:
            SendMessage (hwndClient, WM_MDIICONARRANGE, 0, 0) ;
            return 0 ;

        case IDM_WINDOW_CLOSEALL:       // Attempt to close all children

            EnumChildWindows (hwndClient, CloseEnumProc, 0) ;
            return 0 ;

        default:                    // Pass to active child...

            hwndChild = (HWND) SendMessage (hwndClient,
                                            WM_MDIGETACTIVE, 0, 0) ;
            if (IsWindow (hwndChild))
                SendMessage (hwndChild, WM_COMMAND, wParam, lParam) ;

            break ;         // ...and then to DefFrameProc
        }
        break ;

    case WM_QUERYENDSESSION:
    case WM_CLOSE:                  // Attempt to close all children
```

```
            SendMessage (hwnd, WM_COMMAND, IDM_WINDOW_CLOSEALL, 0) ;

            if (NULL != GetWindow (hwndClient, GW_CHILD))
                return 0 ;

            break ;             // i.e., call DefFrameProc

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
            // Pass unprocessed messages to DefFrameProc (not DefWindowProc)

        return DefFrameProc (hwnd, hwndClient, message, wParam, lParam) ;
}

BOOL CALLBACK CloseEnumProc (HWND hwnd, LPARAM lParam)
{
        if (GetWindow (hwnd, GW_OWNER))         // Check for icon title
            return TRUE ;

        SendMessage (GetParent (hwnd), WM_MDIRESTORE, (WPARAM) hwnd, 0) ;

        if (!SendMessage (hwnd, WM_QUERYENDSESSION, 0, 0))
            return TRUE ;

        SendMessage (GetParent (hwnd), WM_MDIDESTROY, (WPARAM) hwnd, 0) ;
        return TRUE ;
}

LRESULT CALLBACK HelloWndProc (HWND hwnd, UINT message,
                               WPARAM wParam, LPARAM lParam)
{
        static COLORREF  clrTextArray[] = { RGB (0, 0, 0), RGB (255, 0, 0),
                                            RGB (0, 255, 0), RGB ( 0, 0, 255),
                                            RGB (255, 255, 255) } ;
        static HWND      hwndClient, hwndFrame ;
        HDC              hdc ;
        HMENU            hMenu ;
        PHELLODATA       pHelloData ;
        PAINTSTRUCT      ps ;
        RECT             rect ;

        switch (message)
        {
        case WM_CREATE:
                // Allocate memory for window private data
```

```
        pHelloData = (PHELLODATA) HeapAlloc (GetProcessHeap (),
                             HEAP_ZERO_MEMORY, sizeof (HELLODATA)) ;
        pHelloData -> iColor   = IDM_COLOR_BLACK ;
        pHelloData -> clrText = RGB (0, 0, 0) ;
        SetWindowLong (hwnd, 0, (long) pHelloData) ;

            // Save some window handles

        hwndClient = GetParent (hwnd) ;
        hwndFrame  = GetParent (hwndClient) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_COLOR_BLACK:
        case IDM_COLOR_RED:
        case IDM_COLOR_GREEN:
        case IDM_COLOR_BLUE:
        case IDM_COLOR_WHITE:
                // Change the text color

            pHelloData = (PHELLODATA) GetWindowLong (hwnd, 0) ;

            hMenu = GetMenu (hwndFrame) ;

            CheckMenuItem (hMenu, pHelloData -> iColor, MF_UNCHECKED) ;
            pHelloData -> iColor = wParam ;
            CheckMenuItem (hMenu, pHelloData -> iColor, MF_CHECKED) ;

            pHelloData -> clrText = clrTextArray[wParam - IDM_COLOR_BLACK] ;

            InvalidateRect (hwnd, NULL, FALSE) ;
        }
        return 0 ;

    case WM_PAINT:
            // Paint the window

        hdc = BeginPaint (hwnd, &ps) ;

        pHelloData = (PHELLODATA) GetWindowLong (hwnd, 0) ;
        SetTextColor (hdc, pHelloData -> clrText) ;

        GetClientRect (hwnd, &rect) ;

        DrawText (hdc, TEXT ("Hello, World!"), -1, &rect,
```

```
                DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_MDIACTIVATE:
                // Set the Hello menu if gaining focus

            if (lParam == (LPARAM) hwnd)
                SendMessage (hwndClient, WM_MDISETMENU,
                             (WPARAM) hMenuHello, (LPARAM) hMenuHelloWindow) ;

                // Check or uncheck menu item

            pHelloData = (PHELLODATA) GetWindowLong (hwnd, 0) ;
            CheckMenuItem (hMenuHello, pHelloData -> iColor,
                           (lParam == (LPARAM) hwnd) ? MF_CHECKED : MF_UNCHECKED) ;

                // Set the Init menu if losing focus

            if (lParam != (LPARAM) hwnd)
                SendMessage (hwndClient, WM_MDISETMENU, (WPARAM) hMenuInit,
                             (LPARAM) hMenuInitWindow) ;

            DrawMenuBar (hwndFrame) ;
            return 0 ;

        case WM_QUERYENDSESSION:
        case WM_CLOSE:
            if (IDOK != MessageBox (hwnd, TEXT ("OK to close window?"),
                                    TEXT ("Hello"),
                                    MB_ICONQUESTION | MB_OKCANCEL))
                return 0 ;

            break ;         // i.e., call DefMDIChildProc

        case WM_DESTROY:
            pHelloData = (PHELLODATA) GetWindowLong (hwnd, 0) ;
            HeapFree (GetProcessHeap (), 0, pHelloData) ;
            return 0 ;
        }
            // Pass unprocessed message to DefMDIChildProc

        return DefMDIChildProc (hwnd, message, wParam, lParam) ;
}

LRESULT CALLBACK RectWndProc (HWND hwnd, UINT message,
                              WPARAM wParam, LPARAM lParam)
```

```
{
     static HWND  hwndClient, hwndFrame ;
     HBRUSH       hBrush ;
     HDC          hdc ;
     PRECTDATA    pRectData ;
     PAINTSTRUCT  ps ;
     int          xLeft, xRight, yTop, yBottom ;
     short        nRed, nGreen, nBlue ;

     switch (message)
     {
     case WM_CREATE:
          // Allocate memory for window private data

          pRectData = (PRECTDATA) HeapAlloc (GetProcessHeap (),
                                  HEAP_ZERO_MEMORY, sizeof (RECTDATA)) ;

          SetWindowLong (hwnd, 0, (long) pRectData) ;

          // Start the timer going

          SetTimer (hwnd, 1, 250, NULL) ;

          // Save some window handles
          hwndClient = GetParent (hwnd) ;
          hwndFrame  = GetParent (hwndClient) ;
          return 0 ;

     case WM_SIZE:               // If not minimized, save the window size

          if (wParam != SIZE_MINIMIZED)
          {
               pRectData = (PRECTDATA) GetWindowLong (hwnd, 0) ;

               pRectData -> cxClient = LOWORD (lParam) ;
               pRectData -> cyClient = HIWORD (lParam) ;
          }

          break ;          // WM_SIZE must be processed by DefMDIChildProc

     case WM_TIMER:               // Display a random rectangle

          pRectData = (PRECTDATA) GetWindowLong (hwnd, 0) ;
          xLeft     = rand () % pRectData -> cxClient ;
          xRight    = rand () % pRectData -> cxClient ;
          yTop      = rand () % pRectData -> cyClient ;
          yBottom   = rand () % pRectData -> cyClient ;
          nRed      = rand () & 255 ;
```

```
            nGreen    = rand () & 255 ;
            nBlue     = rand () & 255 ;

            hdc = GetDC (hwnd) ;
            hBrush = CreateSolidBrush (RGB (nRed, nGreen, nBlue)) ;
            SelectObject (hdc, hBrush) ;

            Rectangle (hdc, min (xLeft, xRight), min (yTop, yBottom),
                max (xLeft, xRight), max (yTop, yBottom)) ;

            ReleaseDC (hwnd, hdc) ;
            DeleteObject (hBrush) ;
            return 0 ;

        case WM_PAINT:              // Clear the window

            InvalidateRect (hwnd, NULL, TRUE) ;
            hdc = BeginPaint (hwnd, &ps) ;
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_MDIACTIVATE:        // Set the appropriate menu
            if (lParam == (LPARAM) hwnd)
                SendMessage (hwndClient, WM_MDISETMENU, (WPARAM) hMenuRect,
                             (LPARAM) hMenuRectWindow) ;
            else
                SendMessage (hwndClient, WM_MDISETMENU, (WPARAM) hMenuInit,
                             (LPARAM) hMenuInitWindow) ;

            DrawMenuBar (hwndFrame) ;
            return 0 ;

        case WM_DESTROY:
            pRectData = (PRECTDATA) GetWindowLong (hwnd, 0) ;
            HeapFree (GetProcessHeap (), 0, pRectData) ;
            KillTimer (hwnd, 1) ;
            return 0 ;
        }
            // Pass unprocessed message to DefMDIChildProc

        return DefMDIChildProc (hwnd, message, wParam, lParam) ;
}
```

MDIDEMO.RC（摘录）

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"
```

```
//////////////////////////////////////////////////////////////////////////////
// Menu

MDIMENUINIT MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "New &Hello",              IDM_FILE_NEWHELLO
        MENUITEM "New &Rectangle",          IDM_FILE_NEWRECT
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                   IDM_APP_EXIT
    END
END

MDIMENUHELLO MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "New &Hello",              IDM_FILE_NEWHELLO
        MENUITEM "New &Rectangle",          IDM_FILE_NEWRECT
        MENUITEM "&Close",                  IDM_FILE_CLOSE
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                   IDM_APP_EXIT
    END
    POPUP "&Color"
    BEGIN
        MENUITEM "&Black",                  IDM_COLOR_BLACK
        MENUITEM "&Red",                    IDM_COLOR_RED
        MENUITEM "&Green",                  IDM_COLOR_GREEN
        MENUITEM "B&lue",                   IDM_COLOR_BLUE
        MENUITEM "&White",                  IDM_COLOR_WHITE
    END
    POPUP "&Window"
    BEGIN
        MENUITEM "&Cascade \ tShift + F5",  IDM_WINDOW_CASCADE
        MENUITEM "&Tile \ tShift + F4",     IDM_WINDOW_TILE
        MENUITEM "Arrange &Icons",          IDM_WINDOW_ARRANGE
        MENUITEM "Close &All",              IDM_WINDOW_CLOSEALL
    END
END

MDIMENURECT MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
```

```
        MENUITEM "New &Hello",                  IDM_FILE_NEWHELLO
        MENUITEM "New &Rectangle",              IDM_FILE_NEWRECT
        MENUITEM "&Close",                      IDM_FILE_CLOSE
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                       IDM_APP_EXIT
    END
    POPUP "&Window"
    BEGIN
        MENUITEM "&Cascade \ tShift + F5",      IDM_WINDOW_CASCADE
        MENUITEM "&Tile \ tShift + F4",         IDM_WINDOW_TILE
        MENUITEM "Arrange &Icons",              IDM_WINDOW_ARRANGE
        MENUITEM "Close &All",                  IDM_WINDOW_CLOSEALL
    END
END

//////////////////////////////////////////////////////////////////////////////
// Accelerator

MDIDEMO ACCELERATORS DISCARDABLE
BEGIN
    VK_F4,          IDM_WINDOW_TILE,        VIRTKEY, SHIFT, NOINVERT
    VK_F5,          IDM_WINDOW_CASCADE,     VIRTKEY, SHIFT, NOINVERT
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by MDIDemo.rc

#define IDM_FILE_NEWHELLO               40001
#define IDM_FILE_NEWRECT                40002
#define IDM_APP_EXIT                    40003
#define IDM_FILE_CLOSE                  40004
#define IDM_COLOR_BLACK                 40005
#define IDM_COLOR_RED                   40006
#define IDM_COLOR_GREEN                 40007
#define IDM_COLOR_BLUE                  40008
#define IDM_COLOR_WHITE                 40009
#define IDM_WINDOW_CASCADE              40010
#define IDM_WINDOW_TILE                 40011
#define IDM_WINDOW_ARRANGE              40012
#define IDM_WINDOW_CLOSEALL             40013
```

MDIDEMO 支持两种类型非常简单的文档窗口：第一种窗口在它的客户区中央显示“Hello, World!”，另一种窗口显示一系列随机矩形（在源代码清单和标识符名称中，它们分别叫做“Hello”文档和“Rect”文档）。这两种文档窗口的菜单不同，显示“Hello,

World!"的文档窗口有一个允许用户修改文本颜色的菜单。

19.2.1　三个菜单

现在让我们先看看 MDIDEMO.RC 资源描述文件，它定义了程序使用的三个菜单模板。

当文档窗口不存在时，程序显示 MdiMenuInit 菜单，这个菜单只允许用户创建新文档或退出程序。

MdiMenuHello 菜单与显示"Hello, World!"的文档窗口相关联。File 子菜单允许用户打开任何一类新文档、关闭活动文档或退出程序。Color 子菜单允许用户设置文本颜色。Window 子菜单包括以平铺或者层叠的方式安排文档窗口、安排文档图标或关闭所有窗口等选项，这个子菜单也列出了它们创建的所有文档窗口。

MdiMenuRect 菜单与随机矩形文档相关联。除了不包含 Color 子菜单外，它与 MdiMenuHello 菜单一样。

照例 RESOURCE.H 头文件定义所有的菜单标识符。另外，以下三个常量定义在 MDIDEMO.C 中：

```
# define INIT_MENU_POS      0
# define HELLO_MENU_POS     2
# define RECT_MENU_POS      1
```

这些标识符说明每个菜单模板中 Windows 子菜单的位置。程序需要这些信息来通知客户窗口文档列表应出现在哪里。当然，MdiMenuInit 菜单没有 Windows 子菜单，所以如前所述，文档列表应附加在第一个子菜单中(位置 0)。不过，实际上永远不会在此看到文档列表(在后面讨论此程序时，您可以发现这样做的原因)。

定义在 MDIDEMO.C 中的 IDM_FIRSTCHILD 标识符不对应于菜单项，它与出现在 Windows 子菜单上的文档列表中的第一个文档窗口相关联。这个识别符的值应当大于所有其他菜单 ID 的值。

19.2.2　程序初始化

在 MDIDEMO.C 中，WinMain 是从注册框架窗口和两个子窗口的窗口类开始的。窗口过程是 FrameWndProc、HelloWndProc 和 RectWndProc。一般来说，这些窗口类应该与不同的图标相关联。为了简单起见，我们将标准 IDI_APPLICATION 图标用于框架窗口和子窗口。

注意，我们已经将框架窗口类的 WNDCLASS 结构的 hbrBackground 字段定义为 COLOR_APPWORKSPACE 系统颜色。由于框架窗口的客户区被客户窗口所覆盖，并且客户窗口具有这种颜色，所以上面的定义不是完全必要的。但是，在最初显示框架窗口时，

使用这种颜色似乎要好一些。

这三种窗口类中的 lpszMenuName 字段都设置为 NULL。对 Hello 和 Rect 子窗口类来说，这是很自然的。对于框架窗口类，我在创建框架窗口时在 CreateWindow 函数中给出菜单句柄。

Hello 和 Rect 子窗口的窗口类将 WNDCLASS 结构中的 cbWndExtra 字段置为非零值来为每个窗口分配额外空间，这个空间将用于存储指向一个内存块的指针（HELLODATA 和 RECTDATA 结构的大小定义在 MDIDEMO.C 的开始处），这个内存块被用于存储每个文档窗口特有的信息。

下一步，WinMain 使用 LoadMenu 加载三个菜单，并把它们的句柄存储到全局变量中。调用三次 GetSubMenu 函数可以获得 Windows 子菜单（文档列表将加在它上面）的句柄，同样也会把这些函数存储到全局变量中。LoadAccelerators 函数加载加速键表。

在 WinMain 中调用 CreateWindow 创建框架窗口。在 FrameWndProc 中处理 WM_CREATE 消息期间，框架窗口创建客户窗口。这项操作涉及到再一次调用函数 CreateWindow。窗口类被设置为 MDICLIENT，它是预先注册的 MDI 客户区窗口类。在 Windows 中许多对 MDI 的支持被放入了 MDICLIENT 窗口类中。客户区窗口过程作为框架窗口和不同文档窗口的中间层。当调用 CreateWindow 创建客户区窗口时，最后一个参数必须被设置为指向 CLIENTCREATESTRUCT 类型结构的指针。这个结构有两个字段，描述如下：

- hWindowMenu 是要加入文档列表的子菜单的句柄。在 MDIDEMO 中，它是 hMenuInitWindow，是在 WinMain 期间获得的。后面将看到如何修改此菜单。
- idFirstChild 是与文档列表中的第一个文档窗口相关联的菜单 ID。它就是 IDM_FIRSTCHILD。

再让我们回过头来看看 WinMain。MDIDEMO 显示新创建的框架窗口，并进入消息循环。消息循环与正常的循环稍有不同：在调用 GetMessage 从消息队列中获得消息之后，MDI 程序把该消息传送给 TranslateMDISysAccel（以及 TranslateAccelerator，如果像 MDIDEMO 程序一样，程序本身也有菜单加速键的话）。

TranslateMDISysAccel 函数把可能对应特定 MDI 加速键（例如 Ctrl-F6）的击键转换成 WM_SYSCOMMAND 消息。如果 TranslateMDISysAccel 或 TranslateAccelerator 都返回 TRUE（表明某个消息已被这些函数之一转换），就不能调用 TranslateMessage 和 DispatchMessage。

注意传递到 TranslateMDISysAccel 和 TranslateAccelerator 的两个窗口句柄：hwndClient 和 hwndFrame。WinMain 函数通过用 GW_CHILD 参数调用 GetWindow 获得 hwndClient 窗口句柄。

19.2.3 创建子窗口

FrameWndProc 的大部分工作是用于处理通知菜单选择的 WM_COMMAND 消息。与平时一样，FrameWndProc 中 wParam 参数的低位字包含着菜单 ID 号。

在菜单 ID 的值为 IDM_FILE_NEWHELLO 和 IDM_FILE_NEWRECT 的情况下，FrameWndProc 必须创建一个新的文档窗口。这涉及到初始化 MDICREATESTRUCT 结构中的字段(大多数字段对应于 CreateWindow 的参数)，并将消息 WM_MDICREATE 发送给客户窗口，消息的 lParam 参数设置为指向这个结构的指针，然后由客户窗口创建于窗口。(也可以使用 CreateMDIWindow 函数。)

MDICREATESTRUCT 结构中的 szTitle 字段一般是对应于文档的文件名。风格字段设置为窗口风格 WS_HSCROLL、WS_VSCROLL 或这两者的组合，以便在文档窗口中包括滚动条。风格字段也可以包括 WS_MINIMIZE 或 WS_MAXIMIZE，以便在最初时以最小化或最大化状态显示文档窗口。

MDICREATESTRUCT 结构的 lParam 字段为框架窗口和子窗口共享某些变量提供了一种方法。这个域可以设置为含有一个结构的内存块的内存句柄。在子文档窗口的 WM_CREATE 消息期间，lParam 是一个指向 CREATESTRUCT 结构的指针，这个结构的 lpCreateParams 字段是一个指向用于创建窗口的 MDICREATESTRUCT 结构的指针。

客户窗口一旦接收到 WM_MDICREATE 消息就创建一个子文档窗口，并把窗口标题加到用于创建客户窗口的 MDICLIENTSTRUCT 结构中所指定的子菜单的底部。当 MDIDEMO 程序创建它的第一个文档窗口时，这个子菜单就是 MdiMenuInit 菜单中的 File 子菜单。后面将看到这个文档列表将如何被移到 MdiMenuHello 和 MdiMenuRect 菜单的 Windows 子菜单中。

菜单上可以列出 9 个文档，每个文档的前面是带有下划线的数字 1～9。如果创建的文档窗口多于 9 个，则这个清单后跟有 More Windows 菜单项。该项激活带有列表框的对话框，列表框列出了所有文档。这种文档列表的维护是 Windows MDI 支持的最好的特性之一。

19.2.4 关于框架窗口的消息处理

在把注意力转移到子文档窗口之前，我们先继续讨论 FrameWndProc 的消息处理。

当从 File 菜单中选择 Close 时，MDIDEMO 关闭活动子窗口。它通过把 WM_MDIGETACTIVE 消息发送给客户窗口，而获得活动子窗口的句柄。如果子窗口以 WM_QUERYENDSESSION 消息来响应，那么 MDIDEMO 将 WM_MDIDESTROY 消息发送给客户窗口，从而关闭子窗口。

处理 File 菜单中的 Exit 选项只需要框架窗口过程给自己发送一个 WM_CLOSE 消息。

处理 Window 子菜单的 Tile、Cascade 和 Arrange 选项是极容易的，只需把消息 WM_MDITILE、WM_MDICASCADE 和 WM_MDIICONARRANGE 发送给客户窗口。

处理 Close All 选项要稍微复杂一些。FrameWndProc 调用 EnumChildWindows，传送一个引用 CloseEnumProc 函数的指针。此函数把 WM_MDIRESTORE 消息发送给每个子窗口，紧跟着发出 WM_QUERYENDSESSION 和 WM_MDIDESTROY。对图标平铺窗口来说并不就此结束，用 GW_OWNER 参数调用 GetWindow 时，返回的非 NULL 值可以表明这一点。

FrameWndProc 没有处理任何由 Color 菜单中对颜色的选择所导致的 WM_COMMAND 消息，这些消息应该由文档窗口负责处理。因此，FrameWndProc 把所有未经处理的 WM_COMMAND 消息发送到活动子窗口，以便子窗口可以处理那些与它们有关的消息。

框架窗口过程不予处理的所有消息都要送到 DefFrameProc，它在框架窗口过程中取代了 DefWindowProc。即使框架窗口过程捕获了 WM_MENUCHAR、WM_SETFOCUS 或 WM_SIZE 消息，这些消息也要被送到 DefFrameProc 中。

所有未经处理的 WM_COMMAND 消息也必须送给 DefFrameProc。具体地说，FrameWndProc 并不处理任何 WM_COMMAND 消息，即使这些消息是用户在 Windows 子菜单的文档列表中选择文档时产生的(这些选项的 wParam 值是以 IDM_FIRSTCHILD 开始的)。这些消息要传送到 DefFrameProc，并在那里进行处理。

注意框架窗口并不需要维护它所创建的所有文档窗口的窗口句柄清单。如果需要这些句柄(如处理菜单上的 Close All 选项时)，可以使用 EnumChildWindows 得到它们。

19.2.5　子文档窗口

现在看一下 HelloWndProc，它是用于显示“Hello，World!”的子文档窗口的窗口过程。

与用于多个窗口的窗口类一样，所有在窗口过程(或从该窗口过程中调用的任何函数)中定义的静态变量由基于该窗口类的所有窗口共享。

只有对于每个唯一于窗口的数据才必须采用非静态变量的方法来存储。这样的技术要用到窗口属性。另一种方法(我使用的方法)是使用预留的内存空间。可以在注册窗口类时将 WNDCLASS 结构的 cbWndExtra 字段设置为非零值，以便预留这部分内存空间。

MDIDEMO 程序使用这个内存空间来保存一个指针，这个指针指向一块与 HELLODATA 结构大小相同的内存块。在处理 WM_CREATE 消息时，HelloWndProc 分配这块内

存，初始化它的两个字段（它们用于指定当前选中的菜单项和文本颜色），并用 SetWindowLong 将内存指针保存到预留的空间中。

当处理改变文本颜色的 WM_COMMAND 消息（回忆一下，这些消息来自框架窗口过程）时，HelloWndProc 使用 GetWindowLong 获得包含 HELLODATA 结构的内存块的指针。利用这个结构，HelloWndProc 清除原来对菜单项的选择，设置所选菜单项为选中状态，并保存新的颜色。

当窗口变成活动窗口或非活动窗口时，文档窗口过程都会收到 WM_MDIACTIVATE 消息（lParam 的值是否是这个窗口的句柄表明了该窗口是活动的还是非活动的）。您也许还能记起 MDIDEMO 程序中有三个不同的菜单：当无文档时为 MdiMenuInit；当 Hello 文档窗口是活动窗口时为 MdiMenuHello；当 Rect 文档窗口为活动窗口时为 MdiMenuRect。

WM_MDIACTIVATE 消息为文档窗口提供了一个修改菜单的机会。如果 lParam 中含有本窗口的句柄（意味着本窗口将变成活动的），那么 HelloWndProc 就将菜单改为 MdiMenuHello。如果 lParam 中包含另一个窗口的句柄，那么 HelloWndProc 将菜单改为 MdiMenuInit。

HelloWndProc 通过把 WM_MDISETMENU 消息发送给客户窗口来修改菜单，客户窗口通过从当前菜单上删除文档列表并把它添加到一个新的菜单上来处理该消息。这就是文档列表从 MdiMenuInit 菜单（它在创建第一个文档时有效）传送到 MdiMenuHello 菜单中的方法。在 MDI 应用程序中不要用 SetMenu 函数更改菜单。

另一项工作涉及到 Color 子菜单上的复选标记。像这样的程序选项对每个文档来说都是不同的，例如，可以在一个窗口中设置黑色文本，在另一个窗口中设置红色文本。菜单复选标记应能反映出活动窗口中选定的选项。由于这种原因，HelloWndProc 在窗口变成非活动窗口时清除选中菜单项的复选标记，而当窗口变成活动窗口时设置适当菜单项的复选标记。

WM_MDIACTIVATE 的 wParam 和 lParam 值分别是取消激活窗口和激活窗口的句柄。窗口过程得到的第一个 WM_MDIACTIVATE 消息的 lParam 参数被设置为当前窗口的句柄。而当窗口被消除时，窗口过程得到的最后一个消息的 lParam 参数被设置为另一个值。当用户从一个文档切换到另一个文档时，前一个文档窗口收到一个 WM_MDIACTIVATE 消息，其 lParam 参数为第一个窗口的句柄（此时，窗口过程将菜单设置为 MdiMenuInit）；后一个文档窗口收到一个 WM_MDIACTIVATE 消息，其 lParam 参数是第二个窗口的句柄（此时，窗口过程将菜单设置为 MdiMenuHello 或 MdiMenuRect 中适当的那个）。如果所有的窗口都关闭了，剩下的菜单就是 MdiMenuInit。

当用户从菜单中选择 Close 或 Close All 时，FrameWndProc 给子窗口发送一个 WM_QUERYENDSESSION 消息。HelloWndProc 将显示一个消息框，并询问用户是否要关闭窗

口，以此来处理 WM_QUERYENDSESSION 和 WM_CLOSE 消息（在实际的应用程序中，消息框会询问是否需要保存文件）。如果用户表示不关闭窗口，那么窗口过程返回 0。

在 WM_DESTROY 消息期间，HelloWndProc 释放在 WM_CREATE 期间分配的内存块。

所有未经处理的消息必须传送到用于默认处理的 DefMDIChildProc（不是 DefWindowProc）。无论子窗口过程是否使用了这些消息，有几个消息必须被传送给 DefMDIChildProc。这些消息是：WM_CHILDACTIVATE、WM_GETMINMAXINFO、WM_MENUCHAR、WM_MOVE、WM_SETFOCUS、WM_SIZE，以及 WM_SYSCOMMAND。

RectWndProc 与 HelloWndProc 非常相似，但是它比 HelloWndProc 要简单一些（不含菜单选项并且无需用户确认是否关闭窗口），所以这里不对它进行讨论了。但应该注意到，在处理 WM_SIZE 之后 RectWndProc 使用了“break”语句，所以 WM_SIZE 消息被传给 DefMDIChildProc。

19.2.6　结束处理

在 WinMain 中，MDIDEMO 使用 LoadMenu 加载资源描述文件中定义的三个菜单。一般说来，当菜单所在的窗口被清除时，Windows 也要清除与之关联的菜单。对于 Init 菜单，应该清除那些没有联系到窗口的菜单。由于这个原因，MDIDEMO 在 WinMain 的末尾调用了两次 DestroyMenu 来清除 Hello 和 Rect 菜单。

第二十章

多任务和多线程

多任务是一个操作系统可以同时运行多个程序的能力。基本上,操作系统使用一个硬件时钟为同时运行的每个进程分配“时间片”。如果时间片足够小,并且机器也没有由于太多的程序而超负荷,那么在用户看来,所有的这些程序似乎在同时运行着。

多任务并不是什么新的东西。在大型计算机上,多任务是必然的。这些大型机通常有几十甚至几百个终端和它连接,而每个终端用户都应该感觉到他(她)独占了整个计算机。另外,大型机的操作系统通常允许用户“提交后台作业”,这些后台作业可以在用户进行其他工作时,由机器动态完成。

个人计算机上的多任务花了更长的时间才变为现实。但是现在 PC 多任务也被认为是很正常的。我马上就会讨论到,Microsoft Windows 的 16 位版本在一定的程度上支持多任务,Windows 的 32 位版本支持真正的多任务,而且,作为一种额外的收获,它们还支持多线程。

多线程是在一个程序内部实现多任务的能力。程序可以把它自己分隔为单独的执行“线程”,这些线程似乎也同时在运行。这一概念初看起来似乎没有什么用处,但是它可以让程序使用多线程在后台执行长作业,从而使用户不必长时间地无法使用其计算机进行其他工作。当然,有时人们并不那么渴求多任务,找个借口去冲冲凉或者到冰箱去看看总是很不错的!但是,甚至在计算机繁忙的时候,用户也总应该能够在机器上做点什么才是。

20.1 多任务的各种模式

在 PC 的早期,有人曾经提倡未来的多任务,但是大多数的人还是很迷惑:在一个单用户的个人计算机上,多任务有什么用呢?事实证明多任务是用户在对此确实一无所知时需要了解的内容。

20.1.1 DOS下的多任务

最初PC上的Intel 8088微处理器并不是为多任务而设计的。部分原因(我在上一章中讨论过)是内存管理的能力不够强。当启动和结束多个程序时,多任务的操作系统通常需要移动内存块以收集空闲内存。在8088上是不可能透明于应用系统来做到这一点的。

DOS本身对多任务没有太大的帮助,它的设计目标是小型的,并且与应用无关,因此,除了加载程序以及给程序提供对文件系统的访问,它几乎没有提供任何支持。

不过,有创造性的程序员仍然在DOS的早期就找到了一种克服这些缺陷的方法,大多数是使用内存驻留(TSR:terminate-and-stay-resident)程序。有些TSR,比如假脱机打印程序等,通过陷入硬件定时中断来执行真正的后台处理;其他的TSR,诸如SideKick等弹出式工具,可以执行某种类型的任务切换——挂起一个应用,运行弹出式工具。DOS也逐渐有所增强,以便提供对TSR的支持。

一些软件生产商试图在DOS之上构造出任务切换或者多任务外壳(诸如Quarterdeck的DesqView),但是在这些环境中,仅有一个占据了大量的市场,当然,这就是Windows。

20.1.2 非抢占式的多任务

当Microsoft在1985年引入Windows 1.0时,它是最成熟的解决方案,目的是突破DOS的局限。Windows在实模式下运行,但是即使这样,它已经可以在物理内存中移动内存块,这是多任务的前提,虽然移动的方法尚未完全透明于应用程序,但是几乎可以忍受了。

在一种图形窗口环境中,多任务比在一种命令行单用户操作系统中显得更有意义。例如,在经典的命令行UNIX中,可以在命令行之外执行程序,让它们在后台运行,然而,程序的所有显示输出必须被重定向到一个文件中,否则其输出将和用户正在做的事情混在一起。

窗口环境允许多个程序在同一个屏幕上一起运行,前后切换非常容易,并且还可以快速地将数据从一个程序移动到另一个程序中,例如,将绘图程序中创建的图片嵌入由字处理程序编辑的文本文件中。在Windows中,以多种方式支持数据转移,首先是使用剪贴板,后来又使用动态数据交换(DDE),而现在则是通过对象链接和嵌入(OLE)。

不过,早期Windows的多任务实现还不是多用户操作系统中传统的抢占式的时间分片。这些操作系统使用系统时钟周期性地中断一个任务并开始另一个任务。Windows的这些16位版本支持一种被称为"非抢占式的多任务",这种类型的多任务由于Windows基于消息的体系结构而成为可能。通常情况下,一个Windows程序将在内存中

睡眠,直到它收到一个消息为止。这些消息通常是用户的键盘或鼠标输入的直接或间接结果。当处理完消息之后,程序将控制返回给 Windows。

Windows 的 16 位版本不会绝对地基于一个计时器脉冲将控制从一个 Windows 程序切换到另一个,任何的任务切换都发生在当程序完成对消息的处理后将控制返回给 Windows 时。这种非抢占式的多任务也被称为"合作式的多任务",因为它要求来自应用程序方面的一些合作。一个 Windows 程序,如果要花很长一段时间来处理消息的话,那么它可能会阻碍整个系统的运行

虽然非抢占式的多任务是 16 位 Windows 的一般规则,但仍然出现了某些形式的抢占式多任务。Windows 使用抢占式多任务来运行 DOS 程序,而且,为了实现多媒体,还允许动态链接库接收硬件计时器中断。

16 位 Windows 包括几个功能特性来帮助程序员解决(或者,至少可以说是对付)非抢占式多任务中的局限,最显著的当然是时钟式鼠标光标。当然,这并非一种解决方案,而仅仅是让用户知道一个程序正在忙于处理一件长作业,因而系统将在一段时间内不可使用。另一个部分的解决方案是 Windows 计时器,它允许程序周期性地接收消息并完成一些工作。计时器通常用于时钟应用程序和动画。

针对抢占式多任务的另一种解决方案是 PeekMessage 函数调用,我们曾在第五章中的 RANDRECT 程序里看到过。一个程序通常使用 GetMessage 调用从它的消息队列中检索下一个消息,不过,如果在消息队列中没有消息,那么 GetMessage 不会返回,一直到出现一个消息为止。而另一方面,PeekMessage 将控制返回程序,即使没有等待的消息。这样,一个程序可以执行一个长作业,并在代码中混入 PeekMessage 调用。只要没有这个程序或其他任何程序的挂起消息,那么这个长作业将继续运行。

20.1.3 PM 和串行化的消息队列

Microsoft 在一种半 DOS/半 Windows 的环境下实现多任务的第一个尝试(和 IBM 合作)是 OS/2 和显示管理器(PM:Presentation Manager)。虽然 OS/2 明确地支持抢占式多任务,但是这种抢占式似乎并未在显示管理器中得以实现。问题在于 PM 串行化来自键盘和鼠标的用户输入消息。这意味着,直到前一个用户输入消息被完全处理以后,PM 不会将一个键盘或者鼠标消息传送给程序。

尽管键盘和鼠标消息只是一个 PM(或者 Windows)程序可以接收的许多消息中的几个,大多数的其他消息都是键盘或者鼠标事件的结果。例如,菜单命令消息是用户使用键盘或者鼠标进行菜单选择的结果。在处理菜单命令消息时,键盘或者鼠标消息并未完全被处理。

串行化消息队列的主要原因是允许用户的预"键入"和预"按下"。如果一个键盘或鼠标消息导致输入焦点从一个窗口切换到另一个窗口,那么接下来的键盘消息应该进

入拥有新的输入焦点的窗口中。因此，系统并不知道将下一个用户输入消息发送到何处，直到前一个消息被处理完以后。

目前的共识是不应该让一个应用系统有可能占用整个系统，而这要求一个非串行化的消息队列，32 位版本的 Windows 支持这种消息队列。如果一个程序正在忙于处理一项长作业，那么您可以将输入焦点切换到另一个程序中。

20.1.4　多线程解决方案

我讨论 OS/2 的显示管理器，只是因为它是第一个为早期的 Windows 程序员（比如我自己）介绍多线程的环境。有趣的是，PM 实现多线程的局限为程序员提供了应该如何构造多线程程序的必要线索。即使这些局限在 32 位的 Windows 中已经有了大量的减少，但是从更有限的环境中学到的经验仍然是非常有效的。因此，让我们继续。

在一种多线程环境中，程序可以将它们自己分隔为同时运行的片（叫做执行的线程）。对线程的支持是解决 PM 中存在的串行化消息队列的最好方法，并且在 Windows 中线程有更实际的意义。

就代码来说，一个线程简单地被表示为可能调用程序中其他函数的函数。程序从其主线程开始执行，这个主线程是在传统的 C 程序中叫做“main 的函数”，而在 Windows 中是 WinMain。一旦运行起来，程序可以通过在系统调用 CreateThread 中指定初始线程函数的名称来创建新的线程的执行。操作系统在线程之间抢占式地切换控制，和它在进程之间切换控制的方法非常类似。

在 OS/2 的显示管理器中，每个线程可以创建一个消息队列，也可以不创建。如果希望从线程创建窗口，那么一个 PM 线程必须创建消息队列。否则，如果只是进行许多的数据处理或者图形输出，那么线程不需要创建消息队列。因为无消息队列的进程不处理消息，所以它们将不会挂起系统。唯一的限制是一个无消息队列线程无法向一个消息队列线程中的窗口发送消息，或者调用任何发送消息的函数（不过，它们可以将消息传送给消息队列线程）。

这样，PM 程序员学会了如何将它们的程序分隔为一个消息队列线程（在其中创建所有的窗口并处理传送给窗口的消息）和一个或多个无消息队列线程，在其中执行长的后台任务。PM 程序员还了解到“1/10 秒规则”，大体上，程序员被告知，一个消息队列线程处理任何消息都不应该超过 1/10 秒，任何花费更长时间的事情都应该在另一个线程中完成。如果所有的程序员都遵循这一规则，那么将没有 PM 程序会将系统挂起超过 1/10 秒。

20.1.5　多线程体系结构

我已经说过，PM 的局限为程序员理解如何在运行于图形环境中的程序里使用多个线程

的执行提供了必要的线索。因此,在这里我将为您的程序建议一种体系结构:主线程创建程序所需的所有窗口,并在其中包括所有的窗口过程,以便处理这些窗口的所有消息;其他的所有线程只进行一些后台处理,除了和主线程通讯,它们不和用户进行交流。

考虑这种体系结构的一种方法是:主线程处理用户输入(和其他消息),并创建进程中的其他线程,这些附加的线程完成与用户无关的任务。

换句话说,程序的主线程是一个总头目,而其他线程是总头目的职员。总头目将大的工作分配给他(她)的职员,而他(她)保持和外部世界的联系。因为他们仅仅是职员,所以其他线程不会举行他们自己的记者招待会。他们会认真地完成自己的工作,将结果报告给总头目,并等待他们的下一个任务。

一个程序中的线程是同一进程的不同部分,因此它们共享进程的资源,如内存和打开的文件。因为线程共享程序的内存,所以它们还共享静态变量。然而,每个线程都有它们自己的堆栈,因此动态变量对每个线程是唯一的。每个线程还有它自己的处理器状态(和数学协处理器状态),这个状态在进行线程切换期间被保存和恢复。

20.1.6　线程间的"争吵"

正确地设计、编程和调试一个复杂的多线程应用程序显然是 Windows 程序员可能遇到的最困难的工作之一。因为抢占式多任务系统可以在任何时刻中断一个线程,并将控制切换到另一个线程中,在两个线程之间可能有无法预料的随机交互的情况。

多线程程序中的一个常见的错误被称为"竞争条件",这发生在程序员假设一个线程在另一个线程需要某数据之前已经完成了某些处理(如准备数据)的时候。为了帮助协调线程的活动,操作系统要求各种形式的同步。一种是信号量,它允许程序员在代码中的某一点阻止一个线程的执行,直到另一个线程发信号让它继续为止。类似于信号量的是"临界区",它是代码中不可中断的部分。

但是信号量还可能引入称为"死锁"的常见线程相关的错误,这发生在两个线程互相阻止了另一个的执行,而继续执行的唯一办法又是它们继续向前运行。

幸运的是,32 位程序比 16 位程序更能抵抗线程所涉及的某些问题。例如,假定一个线程执行下面的简单语句:

```
lCount++ ;
```

其中 lCount 是由其他线程使用的一个长型的 32 位全局变量,C 语言中的这个语句被编译为两条机器代码指令,第一条将变量的低 16 位加 1,而第二条指令将任何可能的进位加到高 16 位上。假定操作系统在这两个机器代码指令之间中断了线程。如果 lCount 在第一条机器代码指令之前是 0x0000FFFF,那么 lCount 在线程被中断时为 0,而这正是另一个线程将看到的值。只有当线程继续执行时,lCount 才会增加到正确的值 0x00010000。

这是那些偶尔会导致操作问题的错误之一。在16位程序中，解决此问题正确的方法是将语句包含在一个临界区中，在这期间线程不会被中断。然而，在一个32位程序中，该语句是正确的，因为它被编译为一条机器代码指令。

20.1.7 Windows的优点

32位Windows版本(包括Windows NT和Windows 98)有一个所希望的消息队列。这种实现似乎非常好：如果一个程序正在花费一段长时间处理一个消息，那么鼠标位于该程序的窗口上时，鼠标光标将呈现为一个时钟；但是当将鼠标移到另一个程序的窗口上时，鼠标光标将变为正常的箭头形状。只需按一下就可以将另一个窗口提到前台。

然而，用户仍然不能使用正在完成大作业的那个程序，因为大作业会阻止程序接收其他消息，这不是所希望的。一个程序应该总是对消息开放的，所以需要使用次级线程。

在Windows NT和Windows 98中，消息队列线程和无消息队列线程之间没有什么区别，每个线程在创建时都会有它自己的消息队列，从而减少了PM程序中关于线程的一些不便的规则。(然而，在大多数情况下，您仍然想通过一条线程的消息过程处理输入，而将长作业传递给那些不包含窗口的线程，这种结构几乎总是最容易理解的，我们将看到这一点。)

还有更好的事情：Windows NT和Windows 98中有个函数允许线程杀死同一进程中的另一个线程。当开始编写多线程代码时，您将会发现这种功能在有时是很方便的。OS/2的早期版本没有“杀死线程”的函数。

最后的好消息(至少对这里的话题是好消息)是Windows NT和Windows 98实现了一些被称为“线程局部存储(TLS：thread local storage)”的功能。为了理解这一点，回顾一下我在前面提到过的，静态变量(对一个函数来说，既是全局又是局部变量)在线程之间是被共享的，因为它们位于进程的数据存储空间中。动态变量(对一个函数来说总是局部变量)对每一个线程则是唯一的，因为它们占据堆栈上的空间，而每个线程都有它自己的堆栈。

有时让两个或多个线程使用相同的函数，而让这些线程使用唯一于线程的静态变量，那会带来很大便利。这就是线程局部存储，其中涉及一些Windows函数调用，但是Microsoft还为C编译器进行扩展，使线程局部存储的使用更透明于程序员。

20.1.8 新的！先进的！有线程了！

既然已经介绍了线程的现状，让我们来展望一下线程的未来。有时，存在一种过去使用操作系统提供的每一种功能特性的趋势。最坏的情况是，当您的老板走到您的桌前并说：“我听说这种新的某某东西非常火。让我们在自己的程序中用一些这个某某东西

吧。”然后您将花费一个星期的时间，试图去了解这个某某东西可以如何有益于您的应用程序。

应该注意的是，在并不需要多线程的应用程序系统中添加多线程是没有任何意义的。如果您的程序显示时钟光标的时间太长，或者如果它使用 PeekMessage 调用来避免时钟光标，那么请重构您的程序，使用多线程可能会是一个好主意。否则，您是在为难自己，并可能会在代码中引入新的错误。

在某些情况下，时钟光标可能是完全适当的。我在前面提到过“1/10 秒规则”，而将一个大文件加载到内存中可能会花费多于 1/10 秒的时间，这是否意味着文件加载例程应该在分离的线程中实现呢？没有必要。当用户命令一个程序打开文件时，他（她）通常想立即完成该操作。将文件加载例程放在分离的线程中只会增加额外的负担。完全不值得这样做，即使您想向您的朋友夸耀您在编写多线程程序！

20.2 Windows 多线程

创建新的执行线程的 API 函数是 CreateThread，它的语法如下：

```
hThread = CreateThread (&security_attributes, dwStackSize, ThreadProc,
                        pParam, dwFlags, &idThread) ;
```

第 1 个参数是指向 SECURITY_ATTRIBUTES 类型的结构的指针。在 Windows 98 中忽略该参数。在 Windows NT 中，它被置为 NULL。第 2 个参数是用于新线程的初始堆栈大小，默认值为 0。在任何情况下，Windows 根据需要动态延长堆栈的大小。

CreateThread 的第 3 个参数是指向线程函数的指针。名称可以是任意的，但是必须有如下语法：

```
DWORD WINAPI ThreadProc (PVOID pParam) ;
```

CreateThread 的第 4 个参数为传递给 ThreadProc 的参数。这样主线程和次级线程就可以共享数据。

CreateThread 的第 5 个参数通常为 0，但当创建的线程不马上执行时为标志 CREATE_SUSPENDED。线程将挂起直到调用 ResumeThread。第 6 个参数是一个指针，指向接受线程 ID 的值的变量。

大多数 Windows 程序员喜欢用在 PROCESS.H 头文件中声明的 C 运行时的库函数 _beginthread。它的语法如下：

```
hThread = _beginthread (ThreadProc, uiStackSize, pParam) ;
```

它更简单，对于大多数应用程序很完美，这个线程函数的语法为：

```
void __cdecl ThreadProc (void * pParam) ;
```

20.2.1 再论随机矩形

程序 20-1 所示的 RNDRCTMT 程序是第五章中的 RANDRECT 程序的多线程版本，您将回忆起 RANDRECT 使用的是 PeekMessage 循环来显示一系列的随机矩形。

程序 20-1　RNDRCTMT 程序

RNDRCTMT.C

```
/*------------------------
   RNDRCTMT.C——Displays Random Rectangles
               (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include <process.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

HWND  hwnd ;
int   cxClient, cyClient ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("RndRctMT") ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }
```

```
     hwnd = CreateWindow (szAppName, TEXT ("Random Rectangles"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

VOID Thread (PVOID pvoid)
{
     HBRUSH  hBrush ;
     HDC     hdc ;
     int     xLeft, xRight, yTop, yBottom, iRed, iGreen, iBlue ;

     while (TRUE)
     {
          if (cxClient != 0 || cyClient != 0)
          {
               xLeft   = rand () % cxClient ;
               xRight  = rand () % cxClient ;
               yTop    = rand () % cyClient ;
               yBottom = rand () % cyClient ;
               iRed    = rand () & 255 ;
               iGreen  = rand () & 255 ;
               iBlue   = rand () & 255 ;

               hdc = GetDC (hwnd) ;
               hBrush = CreateSolidBrush (RGB (iRed, iGreen, iBlue)) ;
               SelectObject (hdc, hBrush) ;

               Rectangle (hdc, min (xLeft, xRight), min (yTop, yBottom),
                               max (xLeft, xRight), max (yTop, yBottom)) ;

               ReleaseDC (hwnd, hdc) ;
               DeleteObject (hBrush) ;
          }
     }
}
LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
```

```
{
     switch (message)
     {
     case WM_CREATE:
          _beginthread (Thread, 0, NULL) ;
          return 0 ;

     case WM_SIZE:
          cxClient = LOWORD (lParam) ;
          cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_DESTROY:
          PostQuitMessage (0) ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在创建多线程的 Windows 程序时，需要在 Project Settings 对话框中做一些修改。选择 C/C++ 选项卡，然后在 Category 组合框中选择 Code Generation。在 Use Run-Time Library 组合框中，可以看到用于 Release 配置的 Single-Threaded 和用于 Debug 配置的 Debug Single-Threaded。将这些分别改为 Multithreaded 和 Debug Multithreaded。这将把编译程序标志改为/MT，它是编译程序在编译多线程的应用程序所需要的。具体地说，编译程序将在.OBJ 文件中插入 LIBCMT.LIB 文件名，而不是 LIBC.LIB。链接程序使用这个名称与运行时库函数链接。

LIBC.LIB 和 LIBCMT.LIB 文件包含 C 库函数，有些 C 库函数包含静态数据。例如，由于 strtok 函数可能被连续地多次调用，所以它在静态内存中保存了一个指针。在多线程程序中，每个线程必须在 strtok 函数中有它自己的静态指针。因此，这个函数的多线程版本稍微不同于单线程的 strtok 函数。

同时请注意，我在 RNDRCTMT.C 中包含了头文件 PROCESS.H，这个文件定义一个名为_beginthread 的函数，它启动一个新的线程。只有定义了_MT 标识符，才会声明这个函数，这是/MT 标志的另一个结果。

在 RNDRCTMT.C 的 WinMain 函数中，由 CreateWindow 返回的 hwnd 值被保存在一个全局变量中，因此 cxClient 和 cyClient 值也可以由窗口过程的 WM_SIZE 消息获得。

窗口过程以最容易的方法调用_beginthread——简单地以线程函数的地址（称为 Thread）作为第一个参数，其他参数使用 0，线程函数返回 VOID 并有一个参数，该参数是一个指向 VOID 的指针。在 RNDRCTMT 中的 Thread 函数不使用这个参数。

在调用了_beginthread 函数之后，线程函数（以及该线程函数可能调用的其他任何函数）中的代码和程序中的其他代码同时运行。两个或者多个线程使用一个进程中的

同一函数，在这种情况下，动态局部变量（存储在堆栈上）对每个线程是唯一的；对进程中的所有线程来说，所有的静态变量都是一样的。这就是窗口过程设置全局的 cxClient 和 cyClient 变量并由 Thread 函数使用的方式。

有时您需要唯一于各个线程的持久数据。通常，这种数据是静态变量，但在 Windows 98 中，您可以使用“线程局部存储”，我将在本章后面进行讨论。

20.2.2 程序设计竞赛的问题

1986 年 10 月 3 日，Microsoft 举行了为期一天的针对计算机出版社的技术编辑和作者的简短的记者招待会，来讨论他们当时的一组语言产品，包括他们的第一个交互式开发环境 QuickBASIC 2.0。当时，Windows 1.0 出现还不到一年，但是没有人知道我们什么时候能得到与该环境类似的东西。（这用了好几年。）这一事件与众不同的部分原因是由 Microsoft 的公关人员所炮制的“Storm the Gates”程序设计竞赛。比尔·盖茨使用 QuickBASIC 2.0，而计算机出版社的人员可以使用他们选择的任何语言产品。

竞赛的问题是从公众所提出的题目中挑选出来的（挑选那些需要大约半小时编程的问题），问题如下：

创建一个包含 4 个窗口的多任务模拟。第 1 个窗口必须显示一系列的递增数，第 2 个必须显示一系列的递增素数，而第 3 个必须显示 Fibonacci 序列（Fibonacci 序列以数字 0 和 1 开始，每下一个数是其前两个数的和——即 0、1、1、2、3、5、8，等等）。这三个窗口应该在数字达到窗口底部时或者进行滚动，或者自己清除自己。第 4 个窗口必须显示任意半径的圆，而程序必须在按下 Escape 键时终止。

当然，在 1986 年 10 月，在 DOS 下运行的这样一个程序最多只能是多任务模拟，而且没有一个竞赛者具有足够的勇气——其中大多数也没有足够的知识——来为 Windows 编写这个程序。再者，如果真要这么做，当然不会只花半小时了！

参加这次竞赛的大多数人编写了一个程序来将屏幕分为 4 个区域，程序中包含一个循环，依次更新每个窗口，然后检查是否按下了 Escape 键。作为 DOS 环境下的传统习惯，程序占用了 100% 的 CPU 处理。

如果在 Windows 1.0 中编程，那么结果将是类似程序 20-2 所示的 MULTI1 的结果。我说“类似”，是因为我编写的程序是 32 位的，但程序结构和相当多的代码——除了变量和函数参数定义以及 Unicode 支持——都是相同的。

程序 20-2　MULTI1 程序

MULTI1.C

```
/*------------------------
   MULTI1.C——Multitasking Demo
```

```
                (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>
#include <math.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int cyChar ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("Multi1") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = NULL ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }
     hwnd = CreateWindow (szAppName, TEXT ("Multitasking Demo"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
```

```
            DispatchMessage (&msg) ;
        }
        return msg.wParam ;
}

int CheckBottom (HWND hwnd, int cyClient, int iLine)
{
        if (iLine * cyChar + cyChar > cyClient)
        {
            InvalidateRect (hwnd, NULL, TRUE) ;
            UpdateWindow (hwnd) ;
            iLine = 0 ;
        }
        return iLine ;
}

// - - - - - - - - - - - - - - - - - - - -
// Window 1: Display increasing sequence of numbers
// - - - - - - - - - - - - - - - - - - - -

LRESULT APIENTRY WndProc1 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        static int  iNum, iLine, cyClient ;
        HDC         hdc ;
        TCHAR       szBuffer[16] ;

        switch (message)
        {
        case WM_SIZE:
            cyClient = HIWORD (lParam) ;
            return 0 ;
        case WM_TIMER:
            if (iNum < 0)
                iNum = 0 ;

            iLine = CheckBottom (hwnd, cyClient, iLine) ;
            hdc = GetDC (hwnd) ;

            TextOut (hdc, 0, iLine * cyChar, szBuffer,
                    wsprintf (szBuffer, TEXT ("%d"), iNum++ )) ;

            ReleaseDC (hwnd, hdc) ;
            iLine++ ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

```
// - - - - - - - - - - - - - - - - - - - - - - - -
// Window 2: Display increasing sequence of prime numbers
// - - - - - - - - - - - - - - - - - - - - - - - -

LRESULT APIENTRY WndProc2 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static int  iNum = 1, iLine, cyClient ;
     HDC         hdc ;
     int         i, iSqrt ;
     TCHAR       szBuffer[16] ;

     switch (message)
     {
     case WM_SIZE:
          cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_TIMER:
          do
          {
               if ( ++ iNum < 0)
                    iNum = 0 ;

               iSqrt = (int) sqrt (iNum) ;

               for (i = 2 ; i <= iSqrt ; i++ )
                    if (iNum % i == 0)
                         break ;
          }
          while (i <= iSqrt) ;

          iLine = CheckBottom (hwnd, cyClient, iLine) ;
          hdc = GetDC (hwnd) ;

          TextOut (hdc, 0, iLine * cyChar, szBuffer,
                   wsprintf (szBuffer, TEXT ("%d"), iNum)) ;
          ReleaseDC (hwnd, hdc) ;
          iLine++ ;
          return 0 ;
     }
     return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// - - - - - - - - - - - - - - - - - - - - - - - -
// Window 3: Display increasing sequence of Fibonacci numbers
// - - - - - - - - - - - - - - - - - - - - - - - -

LRESULT APIENTRY WndProc3 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
```

```
    static int  iNum = 0, iNext = 1, iLine, cyClient ;
    HDC         hdc ;
    int         iTemp ;
    TCHAR       szBuffer[16] ;

    switch (message)
    {
    case WM_SIZE:
        cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_TIMER:
        if (iNum < 0)
        {
            iNum  = 0 ;
            iNext = 1 ;
        }

        iLine = CheckBottom (hwnd, cyClient, iLine) ;
        hdc = GetDC (hwnd) ;

        TextOut (hdc, 0, iLine * cyChar, szBuffer,
                 wsprintf (szBuffer, "%d", iNum)) ;
        ReleaseDC (hwnd, hdc) ;
        iTemp  = iNum ;
        iNum   = iNext ;
        iNext  += iTemp ;
        iLine  ++ ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// - - - - - - - - - - - - - - - - - -
// Window 4: Display circles of random radii
// - - - - - - - - - - - - - - - - - -

LRESULT APIENTRY WndProc4 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static int  cxClient, cyClient ;
    HDC         hdc ;
    int         iDiameter ;

    switch (message)
    {
    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;
```

```
            return 0 ;

        case WM_TIMER:
            InvalidateRect (hwnd, NULL, TRUE) ;
            UpdateWindow (hwnd) ;

            iDiameter = rand() % (max (1, min (cxClient, cyClient))) ;
            hdc = GetDC (hwnd) ;

            Ellipse (hdc, (cxClient - iDiameter) / 2,
                          (cyClient - iDiameter) / 2,
                          (cxClient + iDiameter) / 2,
                          (cyClient + iDiameter) / 2) ;

            ReleaseDC (hwnd, hdc) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
// - - - - - - - - - - - - - - - -
// Main window to create child windows
// - - - - - - - - - - - - - - - -

LRESULT APIENTRY WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        static HWND         hwndChild[4] ;
        static TCHAR *      szChildClass[] = { TEXT ("Child1"), TEXT ("Child2"),
                                               TEXT ("Child3"), TEXT ("Child4") } ;
        static WNDPROC      ChildProc[] = { WndProc1, WndProc2, WndProc3, WndProc4 } ;
        HINSTANCE           hInstance ;
        int                 i, cxClient, cyClient ;
        WNDCLASS            wndclass ;

        switch (message)
        {
        case WM_CREATE:
            hInstance = (HINSTANCE) GetWindowLong (hwnd, GWL_HINSTANCE) ;

            wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
            wndclass.cbClsExtra     = 0 ;
            wndclass.cbWndExtra     = 0 ;
            wndclass.hInstance      = hInstance ;
            wndclass.hIcon          = NULL ;
            wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
            wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
            wndclass.lpszMenuName   = NULL ;

            for (i = 0 ; i < 4 ; i++)
```

```
        {
            wndclass.lpfnWndProc   = ChildProc[i] ;
            wndclass.lpszClassName = szChildClass[i] ;

            RegisterClass (&wndclass) ;

            hwndChild[i] = CreateWindow (szChildClass[i], NULL,
                           WS_CHILDWINDOW | WS_BORDER | WS_VISIBLE,
                           0, 0, 0, 0,
                           hwnd, (HMENU) i, hInstance, NULL) ;
        }
        cyChar = HIWORD (GetDialogBaseUnits ()) ;
        SetTimer (hwnd, 1, 10, NULL) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;

        for (i = 0 ; i < 4 ; i++)
            MoveWindow (hwndChild[i], (i % 2) * cxClient / 2,
                                      (i > 1) * cyClient / 2,
                        cxClient / 2, cyClient / 2, TRUE) ;
        return 0 ;

    case WM_TIMER:
        for (i = 0 ; i < 4 ; i++)
            SendMessage (hwndChild[i], WM_TIMER, wParam, lParam) ;

        return 0 ;

    case WM_CHAR:
        if (wParam == '\x1B')
            DestroyWindow (hwnd) ;

        return 0 ;

    case WM_DESTROY:
        KillTimer (hwnd, 1) ;
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在这个程序里实际上没有什么我们没见过的东西。主窗口创建 4 个子窗口，每个子窗口占据客户区的一个象限。主窗口还设置一个 Windows 计时器，并将 WM_TIMER

消息发送给 4 个子窗口。

通常一个 Windows 程序应该保留足够的信息，以便在 WM_PAINT 消息期间重建其窗口中的内容。MULTI1 没有这么做，既然它绘制和清除窗口的速度如此之快，所以我认为那是不必要的。

WndProc2 中的素数发生器的效率并不很高，但是有效。如果一个数除了 1 和它自身以外没有别的因数，那么这个数就是素数。当然，要检查一个数是否是素数并不要求使用小于被检查数的所有数来除这个数并检查余数，而只需使用所有小于被检查数的平方根的数。平方根计算是引入浮点数的原因，否则，该程序将是完全基于整数的程序。

MULTI1 程序没有什么不好的地方。使用 Windows 计时器是在 Windows 的早期（和当前）版本中模拟多任务的一种好方法，然而，计时器的使用有时限制了程序的速度。如果程序可以在 WM_TIMER 消息处理中更新它的所有窗口而还有时间富余下来的话，那就意味着它并没有充分利用机器资源。

一种可能的解决方案是在单个 WM_TIMER 消息期间进行两次或多次的更新，但是到底多少次呢？这将不得不依赖于机器的速度，而这有很大的可变性。谁都不想编写一个程序只能适应 25 MHz 的 386 或 50 MHz 的 486 或 100 GHz 的 Pentium VII。

20.2.3　多线程解决方案

让我们来看一看关于这个程序设计问题的一种多线程解决方案，如程序 20-3 所示的 MULTI2 显示。

程序 20-3　MULTI2 程序

MULTI2.C

```
/*--------------------
   MULTI2.C——Multitasking Demo
             (c) Charles Petzold, 1998
   --------------------*/

#include <windows.h>
#include <math.h>
#include <process.h>

typedef struct
{
     HWND  hwnd ;
     int   cxClient ;
     int   cyClient ;
```

```
    int     cyChar ;
    BOOL    bKill ;
}
PARAMS, * PPARAMS ;

LRESULT APIENTRY WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    static TCHAR  szAppName[] = TEXT ("Multi2") ;
    HWND          hwnd ;
    MSG           msg ;
    WNDCLASS      wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = NULL ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Multitasking Demo"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
```

```
     return msg.wParam ;
}

int CheckBottom (HWND hwnd, int cyClient, int cyChar, int iLine)
{
     if (iLine * cyChar + cyChar > cyClient)
     {
          InvalidateRect (hwnd, NULL, TRUE) ;
          UpdateWindow (hwnd) ;
          iLine = 0 ;
     }
     return iLine ;
}

// - - - - - - - - - - - - - - - - - - - - -
// Window 1: Display increasing sequence of numbers
// - - - - - - - - - - - - - - - - - - - - -

void Thread1 (PVOID pvoid)
{
     HDC        hdc ;
     int        iNum = 0, iLine = 0 ;
     PPARAMS    pparams ;
     TCHAR      szBuffer[16] ;

     pparams = (PPARAMS) pvoid ;

     while (!pparams -> bKill)
     {
          if (iNum < 0)
             iNum = 0 ;

          iLine = CheckBottom (pparams -> hwnd, pparams -> cyClient,
                               pparams -> cyChar, iLine) ;
          hdc = GetDC (pparams -> hwnd) ;

          TextOut (hdc, 0, iLine * pparams -> cyChar, szBuffer,
                   wsprintf (szBuffer, TEXT ("%d"), iNum++ )) ;

          ReleaseDC (pparams -> hwnd, hdc) ;
          iLine++ ;
     }
     _endthread () ;
}

LRESULT APIENTRY WndProc1 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static PARAMS params ;
```

```
    switch (message)
    {
    case WM_CREATE:
        params.hwnd = hwnd ;
        params.cyChar = HIWORD (GetDialogBaseUnits ()) ;
        _beginthread (Thread1, 0, &params) ;
        return 0 ;

    case WM_SIZE:
        params.cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_DESTROY:
        params.bKill = TRUE ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// - - - - - - - - - - - - - - - - - - - - - - -
// Window 2: Display increasing sequence of prime numbers
// - - - - - - - - - - - - - - - - - - - - - - -

void Thread2 (PVOID pvoid)
{
    HDC         hdc ;
    int         iNum = 1, iLine = 0, i, iSqrt ;
    PPARAMS     pparams ;
    TCHAR       szBuffer[16] ;

    pparams = (PPARAMS) pvoid ;

    while (!pparams -> bKill)
    {
        do
        {
            if (++iNum < 0)
                iNum = 0 ;

            iSqrt = (int) sqrt (iNum) ;

            for (i = 2 ; i <= iSqrt ; i++)
                if (iNum % i == 0)
                    break ;
        }
        while (i <= iSqrt) ;
        iLine = CheckBottom (pparams -> hwnd, pparams -> cyClient,
                             pparams -> cyChar, iLine) ;
```

```
            hdc = GetDC (pparams -> hwnd) ;

            TextOut (hdc, 0, iLine * pparams -> cyChar, szBuffer,
                     wsprintf (szBuffer, TEXT ("%d"), iNum)) ;

            ReleaseDC (pparams -> hwnd, hdc) ;
            iLine ++ ;
        }
        _endthread () ;
}

LRESULT APIENTRY WndProc2 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        static PARAMS params ;

        switch (message)
        {
        case WM_CREATE:
            params.hwnd = hwnd ;
            params.cyChar = HIWORD (GetDialogBaseUnits ()) ;
            _beginthread (Thread2, 0, &params) ;
            return 0 ;

        case WM_SIZE:
            params.cyClient = HIWORD (lParam) ;
            return 0 ;

        case WM_DESTROY:
            params.bKill = TRUE ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// Window 3: Display increasing sequence of Fibonacci numbers
// - - - - - - - - - - - - - - - - - - - - - - - -

void Thread3 (PVOID pvoid)
{
        HDC         hdc ;
        int         iNum = 0, iNext = 1, iLine = 0, iTemp ;
        PPARAMS     pparams ;
        TCHAR       szBuffer[16] ;

        pparams = (PPARAMS) pvoid ;

        while (!pparams -> bKill)
```

```
    }
        if (iNum < 0)
        {
            iNum = 0 ;
            iNext = 1 ;
        }
        iLine = CheckBottom (pparams -> hwnd,   pparams -> cyClient,
                             pparams -> cyChar, iLine) ;

        hdc = GetDC (pparams -> hwnd) ;

        TextOut (hdc, 0, iLine * pparams -> cyChar, szBuffer,
                 wsprintf (szBuffer, TEXT ("%d"), iNum)) ;

        ReleaseDC (pparams -> hwnd, hdc) ;
        iTemp   = iNum ;
        iNum    = iNext ;
        iNext   += iTemp ;
        iLine++ ;
    }
    _endthread () ;
}

LRESULT APIENTRY WndProc3 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static PARAMS params ;

    switch (message)
    {
    case WM_CREATE:
        params.hwnd = hwnd ;
        params.cyChar = HIWORD (GetDialogBaseUnits ()) ;
        _beginthread (Thread3, 0, &params) ;
        return 0 ;

    case WM_SIZE:
        params.cyClient = HIWORD (lParam) ;
        return 0 ;

    case WM_DESTROY:
        params.bKill = TRUE ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// - - - - - - - - - - - - - - - - -
// Window 4: Display circles of random radii
```

```
//- - - - - - - - - - - - - - - - - -

void Thread4 (PVOID pvoid)
{
     HDC        hdc ;
     int        iDiameter ;
     PPARAMS    pparams ;

     pparams = (PPARAMS) pvoid ;

     while (!pparams -> bKill)
     {
          InvalidateRect (pparams -> hwnd, NULL, TRUE) ;
          UpdateWindow (pparams -> hwnd) ;

          iDiameter = rand() % (max (1,
                                min (pparams -> cxClient, pparams -> cyClient))) ;

          hdc = GetDC (pparams -> hwnd) ;

          Ellipse (hdc, (pparams -> cxClient - iDiameter) / 2,
                        (pparams -> cyClient - iDiameter) / 2,
                        (pparams -> cxClient + iDiameter) / 2,
                        (pparams -> cyClient + iDiameter) / 2) ;

          ReleaseDC (pparams -> hwnd, hdc) ;
     }
     _endthread () ;
}

LRESULT APIENTRY WndProc4 (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static PARAMS params ;

     switch (message)
     {
     case WM_CREATE:
          params.hwnd = hwnd ;
          params.cyChar = HIWORD (GetDialogBaseUnits ()) ;
          _beginthread (Thread4, 0, &params) ;
          return 0 ;

     case WM_SIZE:
          params.cxClient = LOWORD (lParam) ;
          params.cyClient = HIWORD (lParam) ;
          return 0 ;

     case WM_DESTROY:
```

```
            params.bKill = TRUE ;
            return 0 ;
        }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

// - - - - - - - - - - - - - - - -
// Main window to create child windows
// - - - - - - - - - - - - - - - -

LRESULT APIENTRY WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HWND        hwndChild[4] ;
    static TCHAR *     szChildClass[] = { TEXT ("Child1"), TEXT ("Child2"),
                                          TEXT ("Child3"), TEXT ("Child4") } ;
    static WNDPROC     ChildProc[] = { WndProc1, WndProc2, WndProc3, WndProc4 } ;
    HINSTANCE          hInstance ;
    int                i, cxClient, cyClient ;
    WNDCLASS           wndclass ;

    switch (message)
    {
    case WM_CREATE:
        hInstance = (HINSTANCE) GetWindowLong (hwnd, GWL_HINSTANCE) ;
        wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
        wndclass.cbClsExtra     = 0 ;
        wndclass.cbWndExtra     = 0 ;
        wndclass.hInstance      = hInstance ;
        wndclass.hIcon          = NULL ;
        wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
        wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
        wndclass.lpszMenuName   = NULL ;

        for (i = 0 ; i < 4 ; i++)
        {
            wndclass.lpfnWndProc   = ChildProc[i] ;
            wndclass.lpszClassName = szChildClass[i] ;

            RegisterClass (&wndclass) ;

            hwndChild[i] = CreateWindow (szChildClass[i], NULL,
                           WS_CHILDWINDOW | WS_BORDER | WS_VISIBLE,
                           0, 0, 0, 0,
                           hwnd, (HMENU) i, hInstance, NULL) ;
        }

        return 0 ;
```

```
    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;

        for (i = 0 ; i < 4 ; i++)
            MoveWindow (hwndChild[i], (i % 2) * cxClient / 2,
                                      (i > 1) * cyClient / 2,
                       cxClient / 2, cyClient / 2, TRUE) ;
        return 0 ;

    case WM_CHAR:
        if (wParam == '\x1B')
            DestroyWindow (hwnd) ;

        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

MULTI2.C 的 WinMain 和 WndProc 函数非常类似于 MULTI1.C 中的同名函数。WndProc 为 4 个窗口注册 4 种窗口类，创建这些窗口，并在 WM_SIZE 消息期间缩放这些窗口。WndProc 的唯一不同是它不再设置 Windows 计时器，也不再处理 WM_TIMER 消息。

MULTI2 中较大的改变是每个子窗口过程通过在 WM_CREATE 消息期间调用_beginthread 函数来创建另一个线程的执行。总的来说，MULTI2 程序有 5 个同时运行的线程的执行，主线程包含主窗口过程和 4 个子窗口过程，其余的 4 个线程使用名为 Thread1、Thread2 等的函数，这 4 个线程负责绘制 4 个窗口。

我在 RNDRCTMT 程序中给出的多线程代码没有使用_beginthread 的第三个参数，这个参数允许一个创建另一个线程的线程在 32 位变量中将信息传递给其他线程。通常，这个变量是一个指针，而且是指向一个结构的指针，这允许创建线程和新线程共享信息，而不必借助于全局变量。您可以看到，在 MULTI2 中没有全局变量。

对于 MULTI2 程序，我在程序头定义了一个名为 PARAMS 的结构和一个名为 PPARAMS 的指向结构的指针，这个结构有 5 个字段——窗口句柄、窗口的宽度和高度、字符的高度，以及名为 bKill 的布尔变量。最后的结构字段允许创建进程告知创建的进程何时终止。

让我们来看一看 WndProc1，这是显示增加数序列的子窗口过程。窗口过程变得非常简单，唯一的局部变量是一个 PARAMS 结构。在 WM_CREATE 消息期间，它设置这

个结构的 hwnd 和 cyChar 字段，并调用_beginthread 来创建一个使用 Thread1 函数的新线程，并传递给新线程一个指向该结构的指针。在 WM_SIZE 消息期间，WndProc1 设置结构的 cyClient 字段，而在 WM_DESTROY 消息期间，它将 bKill 字段设置为 TRUE。Thread1 函数通过对_endthread 的调用而告结束。这并不是绝对必要的，因为线程将在退出线程函数之后被清除。不过，当退出一个深陷入某种复杂的处理层次的线程时，_endthread是有用的。

Thread1 函数完成在窗口上的实际绘图，并且和程序的其他 4 个线程同时运行。函数接收指向 PARAMS 结构的一个指针，并进入一个 while 循环，每次检查 bKill 是 TRUE 还是 FALSE。如果是 FALSE，那么函数必须进行 MULTI1.C 中的 WM_TIMER 消息期间所作的同样处理——格式化数字、获取设备描述表句柄并使用 TextOut 显示数字。

当您在 Windows 98 中运行 MULTI2 时，将会看到，窗口更新要比在 MULTI1 中快得多，这表明程序在更加有效地利用处理器的资源。在 MULTI1 和 MULTI2 之间还有另一种区别：通常，当您移动或者缩放一个窗口时，默认窗口过程进入一种模态循环，而窗口的所有输出都将停止；在 MULTI2 中，输出将继续。

20.2.4　有问题吗?

似乎 MULTI2 程序并没有达到它应该有的坚固性。我为什么会这样认为呢？让我们来看一看 MULTI2.C 中的一些多线程"缺陷"，以 WndProc1 和 Thread1 为例。

WndProc1 在 MULTI2 的主线程中运行，而 Thread1 与它同时运行，Windows 98 在这两个线程之间进行切换是不可预测的。假定 Thread1 正在运行，并且刚好执行了检查 PARAMS 结构的 bKill 字段是否为 TRUE 的代码。发现不为 TRUE，但是这之后 Windows 98 将控制切换到主线程，这时用户终止了程序，WndProc1 收到一个 WM_DESTROY 消息并将 bKill 参数置为 TRUE。哦，太晚了！操作系统突然切换到 Thread1 中，而该函数会试图获取一个不存在的窗口的设备描述表句柄。

事实证明，这不是一个问题。Windows 98 足够地牢靠，以致图形函数只是失败而已，不会引起任何问题。

正确的多线程编程技术涉及线程同步的使用(尤其是临界区的使用)，我将马上加以详细讨论。大体上，临界区通过对 EnterCriticalSection 和 LeaveCriticalSection 的调用而加以界定。如果一个线程进入一个临界区，那么另一个线程将无法再进入这个临界区。后一个线程被阻止在对 EnterCriticalSection 的调用上，直到第一个线程调用 LeaveCriticalSection 时为止。

在 MULTI2 中的另一个可能存在的问题是，当另外一个线程显示其输出时，主线程可能会收到一个 WM_ERASEBKGND 或 WM_PAINT 消息。这里，使用临界区有助于避免当两个进程试图在同一个窗口上绘图时可能导致的任何问题。但是，经验表明，Win-

dows 98 很恰当地串行化了对图形绘制函数的访问，即当另一个线程正在绘图的时候，一个线程不能在同一个窗口上绘图。

Windows 98 文档提醒说，有一种未进行图形函数串行化的情形，这就是 GDI 对象（如画笔、画刷、字体、位图、区域和调色板等）的使用。有可能发生一个线程破坏了一个对象，而这个对象正在被另一个线程所使用。解决这个问题的方法要求使用临界区，或者最好不要在线程之间共享 GDI 对象。

20.2.5 Sleep 的优点

我曾经提到，我认为对一个多线程程序来说，最好的体系结构是主线程创建程序中的所有窗口，以及所有的窗口过程，并处理所有的窗口消息。其他线程完成后台工作或者长作业。

不过，假设您想在另一个线程中做动画。通常，Windows 中的动画是使用 WM_TIMER 消息来实现的。如果这个线程没有创建窗口，那么它也不会收到这些消息；如果没有计时器，动画又可能会运行得太快。

解决方案是 Sleep 函数。实际上，线程调用 Sleep 函数来自愿地挂起自己，仅有的一个参数是以 ms 计的时间。Sleep 函数调用不会返回，直到指定的时间过去为止。在这段时间内，线程被挂起，并不会被分配给时间片（尽管该线程显然仍然要求在时钟滴答时给予一小段的处理时间，因为系统必须确定线程是否应该重新开始执行）。给 Sleep 一个值为 0 的参数将导致线程交回它尚未使用完的时间片。

当一个线程调用 Sleep 时，只是该线程被挂起指定的时间，系统仍然运行其他的线程，这些线程和挂起线程可以是在同一个进程中，也可以是在另一个进程中。我在第十四章中的 SCRAMBLE 程序中使用了 Sleep 函数，以放慢擦除操作。

通常情况下，您不应该在您的主线程中使用 Sleep 函数，因为这样会减慢对消息的处理速度，但是因为 SCRAMBLE 没有创建任何窗口，所以在那里使用 Sleep 应该是没有问题的。

20.3 线程同步

大约每年一次，在我公寓窗口外面交通繁忙地段的红绿灯会停止工作。结果是造成交通的混乱，虽然轿车一般能避免撞上别的轿车，但是这些车经常挤在一起。

我用术语称两条路相交的十字路口为“临界区”。一辆向南的车和一辆向西的车不可能同时通过一个十字路口而不撞着对方。依赖于交通流量，可以采用不同的方法来解决这个问题。对于视野清楚车辆稀少的路口，可以相信司机有处理的能力。车辆增多可能会要求一个停车标志，而更加繁忙的交通则将要求有红绿灯，红绿灯有助于协调路口的交通（当然，需要这些灯工作正常）。

20.3.1 临界区

在单任务操作系统中，传统的计算机程序不需要红绿灯来帮助协调它们之间的行为。它们在运行时似乎独占了整条路，而且也确实是这样，没有什么会干扰它们的工作。

即使在多任务操作系统中，大多数的程序也似乎各自独立地在运行，但是可能会发生一些问题。例如，两个程序可能会需要同时从同一个文件中读或者对同一文件进行写。在这种情况下，操作系统提供了一种共享文件和记录上锁的技术来帮助解决这个问题。

然而，在支持多线程的操作系统中，情况会变得混乱而且存在潜在的危险。两个或多个线程共享某些数据的情况并不罕见，例如一个线程可以更新一个或者多个变量，而另一个线程可以使用这些变量。有时这会引发一个问题，而有时又不会(记住操作系统将控制从一个线程切换到另一个线程的操作，只能在机器代码指令之间发生。如果只是一个整数被线程共享，那么对这个变量的改变通常发生在单个指令中，因此潜在的问题被最小化了)。

然而，假设线程共享几个变量或者数据结构。通常，这么多个变量或者结构的字段在它们之间必须是一致的。操作系统可以在更新这些变量的过程中间中断一个线程，那么使用这些变量的线程得到的将是不一致的数据。

结果是冲突的发生，并且通常不难想象这样的错误将对程序造成怎样的破坏。我们所需要的是类似于红绿灯的编程技术，以帮助我们对线程交通进行协调和同步，这就是临界区。大体上，一个临界区就是一块不可中断的代码。

有 4 个函数用于临界区。要使用这些函数，您必须定义一个临界区对象，这是一个类型为 CRITICAL_SECTION 的全局变量。例如：

```
CRITICAL_SECTION cs ;
```

这个 CRITICAL_SECTION 数据类型是一个结构，但是其中的字段只能由 Windows 内部使用。这个临界区对象必须首先被程序中的某个线程初始化，通过调用：

```
InitializeCriticalSection (&cs) ;
```

这样就创建了一个名为 cs 的临界区对象。该函数的联机帮助包括下面的警告："临界区对象不能被移动或者复制，进程也不能修改该对象，但必须在逻辑上把它视为不透明的。"这句话，可以被解释为："不要干扰它，甚至不要看它。"

当临界区对象被初始化之后，线程可以通过下面的调用进入临界区：

```
EnterCriticalSection (&cs) ;
```

在这时，线程被认为"拥有"临界区对象。没有两个线程可以同时拥有临界区对象，因

此，如果一个线程进入了临界区，那么下一个使用同一临界区对象调用 EnterCriticalSection 的线程将在函数调用中被挂起。只有当第一个线程通过下面的调用离开临界区时，函数才会返回：

```
LeaveCriticalSection (&cs) ;
```

这时，在 EnterCriticalSection 调用中被挂起的那个线程将拥有临界区，其函数调用也将返回，允许线程继续执行。

当临界区不再被程序所需要时，可以通过调用

```
DeleteCriticalSection (&cs) ;
```

将其删除，该函数释放所有被分配来维护此临界区对象的系统资源。

这种临界区技术涉及“互斥”（此术语在我们继续讨论线程同步时将再次出现）。在任何时刻，只有一个线程能拥有一个临界区。因此，一个线程可以进入一个临界区，设置一个结构的字段，然后退出临界区。另一个使用该结构的线程在访问结构中的字段之前也要先进入该临界区，然后再退出临界区。

注意，您可以定义多个临界区对象，比如 cs1 和 cs2。例如，如果一个程序有 4 个线程，而前两个线程共享一些数据，那么它们可以使用一个临界区对象，而另外两个线程共享一些其他的数据，那么它们可以使用另一个临界区对象。

您在主线程中使用临界区时应该小心。如果从属线程在它自己的临界区中花费了一段长的时间，那么它可能会将主线程阻止过分长的一段时间。从属线程可能只是使用临界区复制该结构的字段到自己的局部变量中。

临界区的一个局限是它们只能用于在同一进程内的线程之间的协调。但是在某些情况下，您需要协调两个不同进程对同一资源的共享（如共享内存等）。在这里您无法使用临界区，但是可以使用一种被称为“互斥对象（mutex object）”的技术。单词“mutex”是人为构造的，代表“mutual exclusion（互斥）”，它在这里精确地表达了我们的目的。我们想防止当一个程序的线程正在更新数据或者使用共享内存或其他资源时被中断。

20.4　事件信号

多线程通常是用于那些必须执行长时间处理的程序。我们可以将一个“大作业”定义为一个可能会违反 1/10 秒规则的程序。显然大作业包括字处理器中的拼写检查、数据库程序中的文件排序，或者索引、电子表格的重计算、打印，甚至包括复杂的绘图。当然，迄今为止我们知道，遵循 1/10 秒规则的最好方法是将大作业分配到另一个线程的执行中。这些额外的线程不会创建窗口，因此它们不受 1/10 秒规则的限制。

通常希望这些额外的线程在完成其任务时能够通知主线程，或者主线程能够停止其他线程正在进行的作业。这就是我们下面将要讨论的。

20.4.1 BIGJOB1 程序

作为一个假想的大作业，我将使用一系列浮点运算，有时这种运算被称为“野蛮的(savage)”性能测试基准。这种计算以一种间接的方式增加一个整数：它求一个数的平方，再对结果取平方根(得到原来的整数)，然后使用 log 和 exp 函数(同样得到原来的整数)，接着使用 atan 和 tan 函数(还是得到原来的整数)，最后对结果加 1。

BIGJOB1 程序如程序 20-4 所示。

程序 20-4　BIGJOB1 程序

BIGJOB1.C

```
/*------------------------
   BIGJOB1.C——Multithreading Demo
              (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include <math.h>
#include <process.h>

#define REP                 1000000

#define STATUS_READY        0
#define STATUS_WORKING      1
#define STATUS_DONE         2

#define WM_CALC_DONE        (WM_USER + 0)
#define WM_CALC_ABORTED     (WM_USER + 1)

typedef struct
{
     HWND hwnd ;
     BOOL bContinue ;
}
PARAMS, * PPARAMS ;

LRESULT APIENTRY WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("BigJob1") ;
     HWND          hwnd ;
```

```
    MSG         msg ;
    WNDCLASS    wndclass ;

    wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
    wndclass.lpszMenuName   = NULL ;
    wndclass.lpszClassName  = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Multithreading Demo"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

void Thread (PVOID pvoid)
{
    double      A = 1.0 ;
    INT         i ;
    LONG        lTime ;
    volatile    PPARAMS pparams ;

    pparams = (PPARAMS) pvoid ;
```

```
    lTime = GetCurrentTime () ;
    for (i = 0 ; i < REP && pparams -> bContinue ; i++ )
        A = tan (atan (exp (log (sqrt (A * A))))) + 1.0 ;

    if (i == REP)
    {
        lTime = GetCurrentTime () - lTime ;
        SendMessage (pparams -> hwnd, WM_CALC_DONE, 0, lTime) ;
    }
    else
        SendMessage (pparams -> hwnd, WM_CALC_ABORTED, 0, 0) ;

    _endthread () ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static INT      iStatus ;
    static LONG     lTime ;
    static PARAMS   params ;
    static TCHAR *  szMessage[] = { TEXT ("Ready (left mouse button begins)"),
                                    TEXT ("Working (right mouse button ends)"),
                                    TEXT ("%d repetitions in %ld msec") } ;
    HDC             hdc ;
    PAINTSTRUCT     ps ;
    RECT            rect ;
    TCHAR           szBuffer[64] ;

    switch (message)
    {
    case WM_LBUTTONDOWN:
        if (iStatus == STATUS_WORKING)
        {
            MessageBeep (0) ;
            return 0 ;
        }

        iStatus = STATUS_WORKING ;

        params.hwnd = hwnd ;
        params.bContinue = TRUE ;

        _beginthread (Thread, 0, &params) ;

        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

    case WM_RBUTTONDOWN:
```

```
            params.bContinue = FALSE ;
            return 0 ;

        case WM_CALC_DONE:
            lTime = lParam ;
            iStatus = STATUS_DONE ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_CALC_ABORTED:
            iStatus = STATUS_READY ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;

            wsprintf (szBuffer, szMessage[iStatus], REP, lTime) ;
            DrawText (hdc, szBuffer, -1, &rect,
                      DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

这是一个相当简单的程序，但是我认为您将看到它如何演示在多线程程序中完成大作业的通用方法。为了使用 BIGJOB1 程序，在窗口的客户区中按下鼠标左键，从而开始野蛮的性能测试计算的 100 万次重复，这在一台 300 MHz 的 Pentium II 机器上将花费 2 秒。当完成计算时，花费的时间将显示在窗口上。当正在进行计算时，您可以通过在客户区中按下鼠标右键来终止它。

让我们来看一看这是如何实现的：

窗口过程保持了一个被叫做 iStatus 的静态变量(该变量可以被设置为在程序开始处定义的三个常数之一，常数以 STATUS 为前缀)，该变量表明程序是否准备好进行一次计算，是否正在进行一次计算，或者是否完成了计算。程序在 WM_PAINT 消息期间使用 iStatus 变量在客户区的中央显示一个适当的字符串。

窗口过程还保持一个静态结构(类型为 PARAMS，也定义在程序的顶部)，该结构是

在窗口过程和其他线程之间的共享数据。结构只有两个字段——hwnd(程序窗口的句柄)和 bContinue,这是一个布尔变量,用于指示线程是否继续计算或者停止。

当您在客户区中按下鼠标左键时,窗口过程将 iStatus 变量置为 STATUS_WORKING,并设置 PARAMS 结构中的两个字段。结构的 hwnd 字段被设置为窗口句柄,当然,bContinue 被设置为 TRUE。

然后窗口过程调用_beginthread 函数。线程函数 Thread 以调用 GetCurrentTime 开始,GetCurrentTime 获取以 ms 为单位的 Windows 启动以来已经运行的时间,然后进入一个 for 循环,重复 100 万次的野蛮测试计算。还要注意,如果 bContinue 被置为了 FALSE,那么线程将退出循环。

在 for 循环之后,线程函数检查它是否确实完成了 100 万次计算。如果是,那么它再次调用 GetCurrentTime 获得所经过的时间,然后使用 SendMessage 向窗口过程发送一个由程序定义的 WM_USER_DONE 消息,经过的时间作为 lParam 参数。如果计算是在未完成之前被终止的(即如果在循环期间 PARAMS 结构的 bContinue 字段变为了 FALSE),那么线程将发送给窗口过程一个 WM_USER_ABORTED 消息,然后线程通过调用_endthread 正常地结束。

在窗口过程中,当您在客户区中按下鼠标右键时,PARAMS 结构的 bContinue 字段被置为 FALSE。这是如何在完成计算之前结束计算的方法。

注意 Thread 中的 pparams 变量定义为 volatile,这种类型限定符向编译器指出变量可能会在实际的程序语句外被修改(例如被另一个线程)。否则,优化的编译器会假设 pparams -> bContinue 不能被 for 循环内的代码修改,没有必要在每层循环中检查变量。volatile 关键字防止这样的优化。

窗口过程处理 WM_USER_DONE 消息时,首先保存经过的时间。对 WM_USER_DONE 和 WM_USER_ABORTED 消息的处理都是通过对 InvalidateRect 的调用产生 WM_PAINT 消息,并在客户区显示一个新的字符串。

提供一个方法(如结构中的 bContinue 字段)允许线程正常终止,这通常是一个好主意。KillThread 函数只有在正常终止线程比较困难时才应该使用,原因是线程可以分配资源,如内存等。如果当线程终止时没有释放所分配的内存,那么内存将仍然是被分配的。线程不是进程:所分配的资源在一个进程的所有线程之间是共享的,因此当线程终止时,资源不会被自动释放。好的程序结构要求一个线程释放由它分配的所有资源。

还应该知道当第二个线程仍在运行时,可以创建第三个线程。如果 Windows 在 SendMessage 调用和_endthread 调用之间,将控制从第二个线程切换到第一个线程,那么窗口过程就可能响应鼠标击键而创建一个新的线程,从而出现了上述的情况。这不是什么问题,但是如果这对您自己的应用程序来说是一个问题,那么您可能会考虑使用临界区来避免线程之间的冲突。

20.4.2 事件对象

BIGJOB1 每当需要执行野蛮测试计算时，就创建一个线程，线程在完成计算之后自动终止。

另一种可选的方法是在程序的整个生存期内保持线程的运行，但是只在必要时才激活它。这是一个应用事件对象的理想情况。

事件对象可以是“有信号的”（也称为“置位的”）。您可以通过下面的调用来创建事件对象：

```
hEvent = CreateEvent (&sa, fManual, fInitial, pszName) ;
```

第一个参数（指向一个 SECURITY_ATTRIBUTES 结构的指针）和最后一个参数（一个事件对象的名字）只有在事件对象被多个进程共享时才有意义。在同一进程中，这些参数通常被设置为 NULL。如果您希望事件对象被初始化为有信号的，那么将 fInitial 参数设置为 TRUE；而如果希望事件对象被初始化为无信号的，则将 fInitial 参数设置为 FALSE。稍后，我将简要地描述 fManual 参数。

为了置位一个现存的事件对象，调用

```
SetEvent (hEvent) ;
```

为了复位一个事件对象，调用

```
ResetEvent (hEvent) ;
```

一个程序通常调用

```
WaitForSingleObject (hEvent, dwTimeOut) ;
```

并且将第二个参数设置为 INFINITE。如果事件对象当前是置位的，那么函数将立即返回，否则，函数将挂起线程，直到事件对象被置位。如果将第二个参数设置为一个以毫秒为单位的超时时间值，这样函数也可能在事件对象被置位之前返回。

如果最初的 CreateEvent 调用的 fManual 参数被设置为 FALSE，那么事件对象将在 WaitForSingleObject 函数返回时自动复位。这种功能通常使得事件对象没有必要使用 ResetEvent 函数。

现在，我们可以来看一看程序 20-5 所示的 BIGJOB2.C 程序。

程序 20-5　BIGJOB2 程序

BIGJOB2.C

```
/*-------------------
   BIGJOB2.C——Multithreading Demo
```

```
                (c) Charles Petzold, 1998
   ----------------------*/

#include <windows.h>
#include <math.h>
#include <process.h>

#define REP                 1000000

#define STATUS_READY        0
#define STATUS_WORKING      1
#define STATUS_DONE         2

#define WM_CALC_DONE        (WM_USER + 0)
#define WM_CALC_ABORTED     (WM_USER + 1)

typedef struct
{
     HWND      hwnd ;
     HANDLE    hEvent ;
     BOOL      bContinue ;
}
PARAMS, * PPARAMS ;

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("BigJob2") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
```

```
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName, TEXT ("Multithreading Demo"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

void Thread (PVOID pvoid)
{
    double      A = 1.0 ;
    INT         i ;
    LONG        lTime ;
    volatile    PPARAMS pparams ;

    pparams = (PPARAMS) pvoid ;

    while (TRUE)
    {
        WaitForSingleObject (pparams -> hEvent, INFINITE) ;

        lTime = GetCurrentTime () ;

        for (i = 0 ; i < REP && pparams -> bContinue ; i++)
            A = tan (atan (exp (log (sqrt (A * A))))) + 1.0 ;
        if (i == REP)
        {
            lTime = GetCurrentTime () - lTime ;

            PostMessage (pparams -> hwnd, WM_CALC_DONE, 0, lTime) ;
        }
        else
            PostMessage (pparams -> hwnd, WM_CALC_ABORTED, 0, 0) ;
    }
```

```
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HANDLE    hEvent ;
     static INT       iStatus ;
     static LONG      lTime ;
     static PARAMS    params ;
     static TCHAR *   szMessage[ ] = { TEXT ("Ready (left mouse button begins)"),
                                       TEXT ("Working (right mouse button ends)"),
                                       TEXT ("%d repetitions in %ld msec") } ;
     HDC              hdc ;
     PAINTSTRUCT      ps ;
     RECT             rect ;
     TCHAR            szBuffer[64] ;

     switch (message)
     {
     case WM_CREATE:
          hEvent = CreateEvent (NULL, FALSE, FALSE, NULL) ;

          params.hwnd = hwnd ;
          params.hEvent = hEvent ;
          params.bContinue = FALSE ;

          _beginthread (Thread, 0, &params) ;

          return 0 ;

     case WM_LBUTTONDOWN:
          if (iStatus == STATUS_WORKING)
          {
               MessageBeep (0) ;
               return 0 ;
          }
          iStatus = STATUS_WORKING ;
          params.bContinue = TRUE ;

          SetEvent (hEvent) ;

          InvalidateRect (hwnd, NULL, TRUE) ;
          return 0 ;

     case WM_RBUTTONDOWN:
          params.bContinue = FALSE ;
          return 0 ;

     case WM_CALC_DONE:
```

```
            lTime = lParam ;
            iStatus = STATUS_DONE ;
            InvalicateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_CALC_ABORTED:
            iStatus = STATUS_READY ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            GetClientRect (hwnd, &rect) ;

            wsprintf (szBuffer, szMessage[iStatus], REP, lTime) ;
            DrawText (hdc, szBuffer, -1, &rect,
                      DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

处理 WM_CREATE 消息时，窗口过程首先创建一个非人工(nonmanual)的被初始化为有信号的(或者说复位的)事件对象，然后它创建线程。

Thread 函数进入一个无限的 while 循环，在循环开始时首先调用 WaitForSingleObject(注意，PARAMS 结构包括一个包含事件对象句柄的字段)。因为事件被初始化为复位的，所以线程被阻止在函数调用中。按下鼠标左键将导致窗口过程调用 SetEvent，这将释放由 WaitForSingleObject 调用产生的第二个线程，并开始野蛮测试计算。当计算完之后，线程再次调用 WaitForSingleObject，但是由于第一次调用已经使事件对象复位，这样，线程将被挂起，直到再次按动鼠标。

在其他方面，程序几乎和 BIGJOB1 完全一样。

20.5　线程局部存储(TLS)

多线程程序中的全局变量(以及任何被分配的内存)被程序中的所有线程共享。在一个函数中的局部静态变量也被使用函数的所有线程共享。一个函数中的局部动态变量是

唯一于各个线程的，因为它们被存储在堆栈上，而每个线程有它自己的堆栈。

有必要存在一些唯一于各个线程的持久性存储。例如，我在本章前面提到过的 C 语言中的 strtok 函数要求这种类型的存储。不幸的是，C 语言不支持这类存储。但是 Windows 中包含有 4 个函数，它们实现了一种技术来做到这一点，并且 Microsoft 对 C 语言的扩展也支持它，这就叫做“线程局部存储”。

下面是 API 工作的方法：

首先，定义一个包含需要唯一于线程的所有数据的结构，例如：

```
typedef struct
{
    int a ;
    int b ;
}
DATA, * PDATA ;
```

主线程调用 TlsAlloc 获得一个索引值：

```
dwTlsIndex = TlsAlloc () ;
```

这个索引值可以存储在一个全局变量中，或者通过参数结构传递给线程函数。

线程函数首先为该数据结构分配内存，并使用上面获得的索引值调用 TlsSetValue：

```
TlsSetValue (dwTlsIndex, GlobalAlloc (GPTR, sizeof (DATA)) ;
```

该函数将一个指针和某个线程及某个线程索引相关联。现在，任何需要使用这个指针的函数（包括最初的线程函数本身）都可以包含如下所示的代码：

```
PDATA pdata ;
...
pdata = (PDATA) TlsGetValue (dwTlsIndex) ;
```

现在函数可以设置或者使用 pdata -> a 和 pdata -> b 了。在线程函数终止以前，它释放分配的内存：

```
GlobalFree (TlsGetValue (dwTlsIndex)) ;
```

当使用该数据的所有线程都终止之时，主线程将释放索引：

```
TlsFree (dwTlsIndex) ;
```

这个过程刚开始可能有点令人迷惑，因此如果能看一看如何实现线程局部存储可能会有帮助（我不知道 Windows 实际上是如何实现的，但下面的方案是可能的）。首先，TlsAlloc 可能只是分配一块内存（长度为 0）并返回一个索引值，即指向这块内存的一个指针。每次使用该索引调用 TlsSetValue 时，通过重新分配将内存块增大 8 个字节。在这 8 个字节中存储的是调用函数的线程 ID（通过 GetCurrentThreadId 来获得）以及传递给

TlsSetValue 函数的指针。TlsSetValue 简单地使用线程 ID 来搜索表，然后返回指针。TlsFree 将释放内存块。所以您看，这可能是一件容易得可以由您自己来实现的事情。不过，有工具已经为您做了也很不错。

Microsoft 对 C 语言的扩展使这更加容易。仅仅在对于每个线程都不同的变量前加上__declspec (thread)，对于任何函数的外部静态变量，则为：

```
__declspec (thread) int iGlobal = 1 ;
```

对于函数内部的静态变量，则为：

```
__declspec (thread) static int iLocal = 2 ;
```

第二十一章

动态链接库

动态链接库(也称为 DLL、动态库或库模块)是 Microsoft Windows 最重要的组成要素之一。大多数与 Windows 相关的磁盘文件要么是程序模块,要么是动态链接库模块。迄今为止,我们都是在编写 Windows 应用程序,现在是尝试编写动态链接库的时候了。许多您已经学会的编写应用程序的规则同样适用于编写这些库模块,但也有一些重要的不同。

21.1 库的基本知识

正如前面所看到的,Windows 应用程序是一个可执行文件,它通常创建一个或几个窗口,并使用消息循环接收用户输入。通常,动态链接库并不能直接执行,也不接收消息。它们是一些独立的文件,其中包含能被程序或其他 DLL 调用来完成一定作业的函数。只有在其他模块调用动态链接库中的函数时,它才发挥作用。

所谓"动态链接",是指 Windows 把一个模块中的函数调用链接到库模块中的实际函数上的过程。在程序开发中,将各种目标模块(.OBJ)、运行库(.LIB)文件,以及经常是已编译的资源(.RES)文件链接在一起,以便创建 Windows 的.EXE 文件,这时的链接是"静态链接"。动态链接与此不同,它发生在运行时刻。

KERNEL32.DLL、USER32.DLL 和 GDI32.DLL,各种驱动程序文件如 KEYBOARD.DRV、SYSTEM.DRV 和 MOUSE.DRV,以及视频和打印机驱动程序都是动态链接库。这些库都被所有 Windows 应用程序使用。

有些动态链接库(如字体文件等)被称为"纯资源"。它们只包含数据(通常是资源的形式),而不包含代码。由此可见,动态链接库的目的之一就是提供能被许多不同的应用程序使用的函数和资源。在常规的操作系统中,只有操作系统本身才包含其他应用程序能够调用来完成某一作业的例程。在 Windows 中,一个模块调用另一个模块函

数的过程被推广了。事实上，编写一个动态链接库，也就是在扩充 Windows。当然，也可认为动态链接库(包括构成 Windows 的那些库例程)是对用户程序的扩充。

尽管一个动态链接库模块可能有其他扩展名(如 .EXE 或 .FON)，但标准扩展名是 .DLL。只有带 .DLL 扩展名的动态链接库才能被 Windows 自动加载。如果文件有其他扩展名，则程序必须显式地使用 LoadLibrary 或者 LoadLibraryEx 函数来加载该模块。

通常您会发现，动态链接库在大型应用程序中最有意义。例如，假设要为 Windows 编写一个由几个不同的程序组成的大型财务软件包，就会发现这些应用程序会使用许多共同的例程。可以把这些公共例程放入一个常规的目标库(带 .LIB 扩展名)中，并在使用 LINK 静态链接时把它们添加到各程序模块中。但这种方法是很浪费的，因为软件包中的每个程序都包含与公共例程相同的代码。而且，如果修改了库中的某个例程，就要重新链接使用此例程的所有程序。然而，如果把这些公共例程放到被称为 ACCOUNT.DLL 的动态链接库中，就可以解决这两个问题。只有库模块才包含所有程序都要用到的例程。这样能为存储文件节省磁盘空间，在同时运行多个应用程序时节省内存，并且可以修改库模块，而不用重新链接各个程序。

动态链接实际上是一种可以独立自存的产品。例如，假设您编写了一系列三维绘图例程，并把它们放入名为 GDI3.DLL 的 DLL 中。如果其他软件开发者对此库很感兴趣，您就可以授权他们将其加入他们的图形程序中。使用多个这样的图形程序的用户只需要一个 GDI3.DLL 文件。

21.1.1　库：一词多义

关于动态链接库所存在的模糊认识，部分原因是由于“库”这个词被用在几种不同的上下文中。除了动态链接库之外，我们也用它来称呼“目标库”或“输入库”。

目标库是带 .LIB 扩展名的文件。在用链接程序进行静态链接时，它的代码就会加到程序的 .EXE 文件中。例如，在 Microsoft Visual C++ 中，与程序链接的常规 C 运行目标库被称为 LIBC.LIB。

输入库是目标库文件的一种特殊形式。像目标库一样，输入库有 .LIB 扩展名，并且被链接程序用来确定源程序代码中的函数调用。但输入库不含代码，而是为链接程序提供信息，以便在 .EXE 文件中建立动态链接时要用到的重定位表。包含在 Microsoft 编译器中的 KERNEL32.LIB、USER32.LIB 和 GDI32.LIB 文件是 Windows 函数的输入库。如果一个程序调用 Rectangle 函数，那么 Rectangle 将告诉 LINK，该函数在 GDI32.DLL 动态链接库中。该信息被记录在 .EXE 文件中，使得程序运行时，Windows 能够和 GDI32.DLL 动态链接库进行动态链接。

目标库和输入库只用在程序开发期间，而动态链接库在运行期间使用。当一个使用动态链接库的程序运行时，该动态链接库必须在磁盘上。当 Windows 需要运行一个

使用了动态链接库的程序而需要加载该库时，库文件必须存储在含有该.EXE 程序的目录下，或者是在当前的目录下，或 Windows 系统目录下，或 Windows 目录下，或者是在通过 MS-DOS 环境中的 PATH 可以访问到的目录下(将此顺序搜索这些目录)。

21.1.2 一个简单的 DLL

虽然动态链接库的整体概念是它们可以被多个应用程序所使用，但您通常最初设计的动态链接库只与一个应用程序相联系，可能是一个 test 程序在使用 DLL。

下面就是我们要做的。我们创建一个名为“EDRLIB.DLL”的 DLL。文件名中的“EDR”代表“简便的绘图例程(easy drawing routines)”。这里的 EDRLIB 只含有一个函数(名称为 EdrCenterText)，但是您还可以将应用程序中其他简单的绘图函数添加进去。应用程序 EDRTEST.EXE 将通过调用 EDRLIB.DLL 中的函数来利用它。

要做到这一点，需要与我们以前所做的略有不同的方法，也包括 Visual C++ 中我们没有介绍过的特性。在 Visual C++ 中“工作空间(workspaces)”和“项目(projects)”不同。项目通常与创建的应用程序(.EXE)或者动态链接库(.DLL)相联系。一个工作空间可以包含一个或多个项目。迄今为止，我们所有的工作空间都只包含一个项目。我们现在就创建一个包含两个项目的工作空间 EDRTEST——一个用于创建 EDRTEST.EXE，而另一个用于创建 EDRLIB.DLL，即 EDRTEST 使用的动态链接库。

现在就开始。在 Visual C++ 中，从 File 菜单选择 New，然后选择 Workspaces 选项卡。(我们以前从来没有选择过。)在 Location 域选择工作空间要保存的目录，然后在 Workspace Name 域输入 EDRTEST，按 Enter 键。

这样就创建了一个空的工作空间。Developer Studio 还创建了一个名为 EDRTEST 的子目录，以及工作空间文件 EDRTEST.DSW(以及其他文件)。

现在让我们在此工作空间里创建一个项目。从 File 菜单选择 New，然后选择 Projects 选项卡。以前您选择 Win32 Application，但现在需要选择 Win32 Dynamic-Link Library。另外，单击单选按钮 Add To Current Workspace，这使得此项目是 EDRTEST 工作空间的一部分。在 Project Name 域输入 EDRLIB，但先不要按 OK 按钮。当您在 Project Name 域输入 EDRLIB 时，Visual C++ 将改变 Location 域，以显示 EDRLIB 作为 EDRTEST 的一个子目录。您不希望如此！在 Location 域，删除 EDRLIB 子目录以便项目创建在 EDRTEST 目录。现在按 OK。屏幕将显示一个对话框，询问您创建什么类型的 DLL。选择 An Empty DLL Project，然后按 Finish。Visual C++ 将创建一个项目文件 EDRLIB.DSP 和一个构造文件 EDRLIB.MAK(如果 Tools Options 对话框的 Build 选项卡中选中了 Export Makefile 选项)。

现在您可以在此项目中加入一些文件。从 File 菜单选择 New，然后选择 Files 选项卡，选择 C/C++ Header File，然后输入文件名 EDRLIB.H。输入程序 21-1 所示的文件

(或者从本书光盘中复制)。再次从 File 菜单中选择 New,然后选择 Files 选项卡,这次选择 C++ Source File,然后输入文件名 EDRLIB.C,继续输入程序 21-1 所示的程序。

程序 21-1　EDRLIB 库

EDRLIB.H

```
/*------------
     EDRLIB.H header file
   ------------*/

#ifdef__cplusplus
#define EXPORT extern "C" __declspec (dllexport)
#else
#define EXPORT __declspec (dllexport)
#endif

EXPORT BOOL CALLBACK EdrCenterTextA (HDC, PRECT, PCSTR) ;
EXPORT BOOL CALLBACK EdrCenterTextW (HDC, PRECT, PCWSTR) ;

#ifdef UNICODE
#define EdrCenterText EdrCenterTextW
#else
#define EdrCenterText EdrCenterTextA
#endif
```

EDRLIB.C

```
/*----------------------------
     EDRLIB.C——Easy Drawing Routine Library module
                (c) Charles Petzold, 1998
   ----------------------------*/

#include windows.h>
#include "edrlib.h"

int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)
{
     return TRUE ;
}

EXPORT BOOL CALLBACK EdrCenterTextA (HDC hdc, PRECT prc, PCSTR pString)
{
     int       iLength ;
     SIZE      size ;

     iLength = lstrlenA (pString) ;
```

```
    GetTextExtentPoint32A (hdc, pString, iLength, &size) ;

    return TextOutA (hdc, (prc -> right - prc -> left - size.cx) / 2,
                          (prc -> bottom - prc -> top - size.cy) / 2,
                     pString, iLength) ;
}

EXPORT BOOL CALLBACK EdrCenterTextW (HDC hdc, PRECT prc, PCWSTR pString)
{
    int     iLength ;
    SIZE    size ;

    iLength = lstrlenW (pString) ;

    GetTextExtentPoint32W (hdc, pString, iLength, &size) ;

    return TextOutW (hdc, (prc -> right - prc -> left - size.cx) / 2,
                          (prc -> bottom - prc -> top - size.cy) / 2,
                     pString, iLength) ;
}
```

这里您可以按 Release 配置,也可以按 Debug 配置来建立 EDRLIB.DLL。之后,RELEASE 和 DEBUG 目录将包含 EDRLIB.LIB(即动态链接库的输入库)和 EDRLIB.DLL(动态链接库本身)。

纵观全书,我们创建的所有程序都可以根据 Unicode 标识符来编译成面向 Unicode 或非 Unicode 字符串。当您创建一个 DLL 时,它应该包括处理字符和字符串的 Unicode 和非 Unicode 版的所有函数。因此,EDRLIB.C 就包含函数 EdrCenterTextA(ANSI 版)和 EdrCenterTextW(宽字符版)。EdrCenterTextA 定义为带有参数 PCSTR(指向 const 串的指针),而 EdrCenterTextW 则定义为带有参数 PCWSTR(指向 const 宽串的指针)。EdrCenterTextA 函数将调用 lstrlenA、GetTextExtentPoint32A 和 TextOutA。EdrCenterTextW 将调用 lstrlenW、GetTextExtentPoint32W 和 TextOutW。如果定义了 UNICODE 标识符,则 EDRLIB.H 将 EdrCenterText 定义为 EdrCenterTextW,否则定义为 EdrCenterTextA。这很像 Windows 头文件。

EDRLIB.H 也包含函数 DllMain,取代了 DLL 中的 WinMain。此函数用于执行初始化和取消初始化,我将在本章的下一节讨论此问题。我们现在所需要的就是从 DllMain 返回 TRUE。

在这两个文件中,最后一点特殊之处就是定义 EXPORT 标识符。DLL 中应用程序使用的函数必须是"导出(exported)"的。这不是任何税务或者商业制度,只是确保函数名添加到 EDRLIB.LIB 的一个关键词(以便连接器在连接使用此函数的应用程序时,能够解析函数名),而且该函数在 EDRLIB.DLL 中也是可视的。EXPORT 标识符包括保存类说明__declspec(dllexport),以及当头文件按 C++ 模块编译时附加的"C"。这将防止编

译程序破坏 C++ 的函数名，而且能够允许 C 和 C++ 程序都能使用这个 DLL。

21.1.3 库入口/出口点

当库首次启动和结束时，我们调用了 DllMain 函数。DllMain 的第一个参数是库的实例句柄。如果您的库使用需要实例句柄（诸如 DialogBox）的资源，那么您应该将 hInstance 保存为一个全局变量。DllMain 的最后一个参数由系统保留。

fdwReason 参数可以是 4 个值之一，说明为什么 Windows 要调用 DllMain 函数。在下面的讨论中，请记住一个程序可以被多次加载，并在 Windows 下一起运行。每当加载一个程序时，它都被认为是一个独立的进程。

fdwReason 的一个值 DLL_PROCESS_ATTACH 表示动态链接库被映射到一个进程的地址空间。链接库可以根据这个线索进行初始化，为以后来自该进程的请求提供服务。例如，这类初始化可能包括内存分配。在一个进程的生命周期内，只有一次对 DllMain 的调用以 DLL_PROCESS_ATTACH 为参数。使用同一 DLL 的其他任何进程都将导致另一个使用 DLL_PROCESS_ATTACH 参数的 DllMain 调用，但这是对新进程的调用。

如果初始化成功，DllMain 应该返回一个非 0 值。返回 0 值将导致 Windows 不运行该程序。

当 fdwReason 的值为 DLL_PROCESS_DETACH 时，意味着进程不再需要 DLL，从而提供给库自己清除自己的机会。在 32 位的 Windows 下，这种处理并不是严格必须的，但这是一种好的编程习惯。

类似地，当以 DLL_THREAD_ATTACH 为 fdwReason 参数调用 DllMain 时，意味着某个进程创建了一个新的线程。当线程中止时，Windows 以 DLL_THREAD_DETACH 为 fdwReason 参数调用 DllMain。请注意，如果动态链接库是在线程被创建之后和一个进程链接的，那么可能会得到一个没有相应的先行 DLL_THREAD_ATTACH 调用的 DLL_THREAD_DETACH 调用。

当使用一个 DLL_THREAD_DETACH 参数调用 DllMain 时，线程仍然存在。动态链接库甚至可以在这个过程期间发送线程消息。但是它不应该使用 PostMessage，因为线程可能在此消息被检索到之前已经退出了。

21.1.4 测试程序

现在让我们在 EDRTEST 工作空间里创建第二个项目，程序名称为 EDRTEST，而且使用 EDRLIB.DLL。在 Visual C++ 中加载 EDRTEST 工作空间时，请从 File 菜单选择 New，然后在 New 对话框中选择 Projects 选项卡。这次选择 Win32 Application，并确保选中 Add To Current Workspace 按钮，输入项目名称 EDRTEST，然后在 Locations 域删除第二个

EDRTEST 子目录。按下 OK,然后在下一个对话框选择 An Empty Project,按 Finish。

从 File 菜单再次选择 New,选择 Files 选项卡,然后选择 C++ Source File。确保 Add To Project 列表框显示的是 EDRTEST 而不是 EDRLIB,输入文件名 EDRTEST.C,然后输入程序 21-2 所示的程序。此程序用 EdrCenterText 函数将客户区中的文本串居中。

程序 21-2　EDRTEST 程序

EDRTEST.C

```
/*-------------------------------------
   EDRTEST.C——Program using EDRLIB dynamic-link library
              (c) Charles Petzold, 1998
   -------------------------------------*/

#include <windows.h>
#include "edrlib.h"

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     static TCHAR  szAppName[] = TEXT ("StrProg") ;
     HWND          hwnd ;
     MSG           msg ;
     WNDCLASS      wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }
```

```
    hwnd = CreateWindow (szAppName, TEXT ("DLL Demonstration Program"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
         TranslateMessage (&msg) ;
         DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    HDC         hdc ;
    PAINTSTRUCT ps ;
    RECT        rect ;

    switch (message)
    {
    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;

        EdrCenterText (hdc, &rect,
                       TEXT ("This string was displayed by a DLL")) ;

        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

注意，为定义 EdrCenterText 函数，EDRTEST.C 包括 EDRLIB.H 头文件，此函数将在 WM_PAINT 消息期间调用。

在编译此程序之前，您可能希望做以下几件事。首先，在 Project 菜单选择 Select

Active Project。这时您将看到 EDRLIB 和 EDRTEST,选择 EDRTEST。在连编此工作空间时,您真正要连编的是程序。另外,在 Project 菜单中,选择 Dependencies,在 Select Project To Modify 列表框中选择 EDRTEST。在 Dependent On The Following Project(s)列表选中 EDRLIB。此操作的意思是:EDRTEST 需要 EDRLIB 动态链接库。以后每次连编 EDRTEST 时,如果必要的话,都将在编译和连接 EDRTEST 之前重新连编 EDRLIB。

从 Project 菜单选择 Settings,单击 General 标签。当您在左边的窗格中选择 EDRLIB 或者 EDRTEST 项目时,如果配置为 Win32 Release,则显示在右边窗格中的 Intermediate Files 和 Output Files 将位于 RELEASE 目录;如果配置为 Win32 Debug,则位于 DEBUG 目录。如果不是,请按此修改。这样可确保 EDRLIB.DLL 与 EDRTEST.EXE 在同一个目录中,而且程序在使用 DLL 时也不会产生问题。

在 Project Setting 对话框中依然选中 EDRTEST,单击 C/C++ 选项卡。按本书的惯例,在 Preprocessor Definitions 中,将 UNICODE 添加到 Debug 配置。

现在您就可以在 Debug 或 Release 配置中连编 EDRTEST.EXE 了。必要时,Visual C++将首先编译和连接 EDRLIB。RELEASE 和 DEBUG 目录都包含 EDRLIB.LIB(输入库)和 EDRLIB.DLL。当 Developer Studio 连接 EDRTEST 时,将自动包含输入库。

了解 EDRTEST.EXE 文件中不包含 EdrCenterText 代码很重要。事实上,要证明运行了 EDRLIB.DLL 文件和 EdrCenterText 函数很简单:运行 EDRTEST.EXE 需要 EDRLIB.DLL。

运行 EDRTEST.EXE 时,Windows 按外部库模块执行固定的函数,其中许多函数都在 Windows 常规动态链接库中。但 Windows 也看到程序从 EDRLIB 调用了函数,因此 Windows 将 EDRLIB.DLL 文件加载到内存中,然后调用 EDRLIB 的初始化例程。EDRTEST 调用 EdrCenterText 函数是动态链接到 EDRLIB 中的函数的。

在 EDRTEST.C 源代码文件中包含 EDRLIB.H 与包含 WINDOWS.H 类似。连接 EDRLIB.LIB 与连接 Windows 输入库(例如 USER32.LIB)类似。当您的程序运行时,它连接 EDLIB.DLL 的方式与连接 USER32.DLL 的方式相同。恭喜您!您已经扩展了 Windows!

在继续之前,我还要对动态链接库说几句:

首先,虽然我们将 DLL 作为 Windows 的延伸,但它也是您的应用程序的延伸。DLL 完成的每件工作对于应用程序来说都是好的。例如,应用程序拥有它分配的全部内存、它创建的全部窗口,以及它打开的所有文件。多个应用程序可以同时使用同一个 DLL,但在 Windows 下,这些应用程序不会相互影响。

多个进程能够共享一个动态链接库中相同的代码。但是,DLL 为每个进程保存的数据都不同。每个进程都为 DLL 使用的全部数据分配了自己的地址空间。我们将在下一节看到,在进程间共享内存还需要额外的工作。

21.1.5 在 DLL 中共享内存

令人兴奋的是，Windows 能够将同时使用同一个动态链接库的应用程序分开。不过，有时却不太令人满意。您可能希望写一个 DLL，其中包含能够被不同应用程序或者同一个程序的不同例程共享的内存。这包括使用共享内存，实际上是一个内存映射文件。

让我们测试一下，这项工作是如何在程序 STRPROG("串程序(string program)")和动态链接库 STRLIB("串库(string library)")中完成的。STRLIB 有三个输出函数被 STRPROG 调用，我们只对此感兴趣，STRLIB 中的一个函数使用了在 STRPROG 定义的回调函数。

STRLIB 是一个动态链接库模块，它保存并排序最多 256 个字符串。在 STRLIB 中，这些串均为大写，并由共享内存维护。利用 STRLIB 的三个函数，STRPROG 能够添加串、删除串，以及从 STRLIB 获得当前的所有串。STRPROG 测试程序有两个菜单项(Enter 和 Delete)，这两个菜单项将激活不同的对话框来添加或删除串。STRPROG 在其客户区列出当前保存在 STRLIB 中的所有串。

下面这个函数在 STRLIB 定义，它将一个串添加到 STRLIB 的共享内存。

```
EXPORT BOOL CALLBACK AddString (pStringIn)
```

参数 pStringIn 是串的指针。串在 AddString 函数中变成大写。如果在 STRLIB 的列表中有一个相同的串，那么此函数将添加一个串的复本。如果成功，AddString 返回 TRUE(非 0)，否则返回 FALSE(0)。如果串的长度为 0，或者不能分配保存串的内存，或者已经保存了 256 个串，则返回值将都是 FALSE。

STRLIB 函数从 STRLIB 的共享内存中删除一个串：

```
EXPORT BOOL CALLBACK DeleteString (pStringIn)
```

另外，参数 pStringIn 是一个串指针。如果有多个相匹配的串，则删除第一个。如果成功，那么 DeleteString 返回 TRUE(非 0)，否则返回 FALSE(0)。返回 FALSE 表明串和长度为 0，或者找不到相匹配的串。

STRLIB 函数使用了调用程序中的一个回调函数，以便列出目前保存在 STRLIB 共享内存中的串：

```
EXPORT int CALLBACK GetStrings (pfnGetStrCallBack, pParam)
```

在调用程序中，回调函数必须像下面那样定义：

```
EXPORT BOOL CALLBACK GetStrCallBack (PSTR pString, PVOID pParam)
```

GetStrings 的参数 pfnGetStrCallBack 指向回调函数。直到回调函数返回 FALSE(0)，GetStrings 将为每个串都调用一次 GetStrCallBack。GetStrings 返回传递给回调函数的串数。

pParam 参数是一个远指针，指向程序员定义的数据。

当然，此程序可以编译成 Unicode 程序，或者在 STRLIB 的支持下，编译成 Unicode 和非 Unicode 应用程序。与 EDRLIB 一样，所有的函数都有“A”和“W”两种版。在内部，STRLIB 按 Unicode 保存所有的串。如果非 Unicode 程序使用了 STRLIB(也就是说，程序将调用 AddStringA、DeleteStringA 和 GetStringsA)，串将在 Unicode 和非 Unicode 之间转换。

与 STRPROG 和 STRLIB 项目相关的工作空间名为 STRPROG。此文件按 EDRTEST 工作空间的方式组合。程序 21-3 显示了创建 STRLIB.DLL 动态链接库所必需的两个文件。

程序 21-3 STRLIB 库

STRLIB.H

```
/* - - - - - - - - - -
   STRLIB.H header file
   - - - - - - - - - - */

# ifdef__cplusplus
# define EXPORT extern "C" __declspec (dllexport)
# else
# define EXPORT __declspec (dllexport)
# endif

    // The maximum number of strings STRLIB will store and their lengths

# define MAX_STRINGS 256
# define MAX_LENGTH 63

    // The callback function type definition uses generic strings

typedef BOOL (CALLBACK * GETSTRCB) (PCTSTR, PVOID) ;

    // Each function has ANSI and Unicode versions

EXPORT BOOL CALLBACK AddStringA (PCSTR) ;
EXPORT BOOL CALLBACK AddStringW (PCWSTR) ;

EXPORT BOOL CALLBACK DeleteStringA (PCSTR) ;
EXPORT BOOL CALLBACK DeleteStringW (PCWSTR) ;

EXPORT int CALLBACK GetStringsA (GETSTRCB, PVOID) ;
EXPORT int CALLBACK GetStringsW (GETSTRCB, PVOID) ;

    // Use the correct version depending on the UNICODE identifier
```

```
# ifdef UNICODE
# define AddString          AddStringW
# define DeleteString       DeleteStringW
# define GetStrings         GetStringsW
# else
# define AddString          AddStringA
# define DeleteString       DeleteStringA
# define GetStrings         GetStringsA
# endif
```

STRLIB.C

```
/*-----------------------
   STRLIB.C——Library module for STRPROG program
             (c) Charles Petzold, 1998
   -----------------------*/

# include <windows.h>
# include <wchar.h>              // for wide-character string functions
# include "strlib.h"

    // shared memory section (requires /SECTION:shared,RWS in link options)

# pragma data_seg ("shared")
int iTotal = 0 ;
WCHAR szStrings [MAX_STRINGS][MAX_LENGTH + 1] = {'\0'} ;
# pragma data_seg ()

# pragma comment(linker,"/SECTION:shared,RWS")

int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)
{
    return TRUE ;
}

EXPORT BOOL CALLBACK AddStringA (PCSTR pStringIn)
{
     BOOL     bReturn ;
     int      iLength ;
     PWSTR    pWideStr ;

         // Convert string to Unicode and call AddStringW

     iLength = MultiByteToWideChar (CP_ACP, 0, pStringIn, -1, NULL, 0) ;
     pWideStr = malloc (iLength) ;
     MultiByteToWideChar (CP_ACP, 0, pStringIn, -1, pWideStr, iLength) ;
     bReturn = AddStringW (pWideStr) ;
     free (pWideStr) ;
```

```
    return bReturn ;
}

EXPORT BOOL CALLBACK AddStringW (PCWSTR pStringIn)
{
    PWSTR  pString ;
    int    i, iLength ;

    if (iTotal == MAX_STRINGS - 1)
        return FALSE ;

    if ((iLength = wcslen (pStringIn)) == 0)
        return FALSE ;

        // Allocate memory for storing string, copy it, convert to uppercase

    pString = malloc (sizeof (WCHAR) * (1 + iLength)) ;
    wcscpy (pString, pStringIn) ;
    _wcsupr (pString) ;

        // Alphabetize the strings

    for (i = iTotal ; i > 0 ; i--)
    {
        if (wcscmp (pString, szStrings[i - 1]) >= 0)
            break ;

        wcscpy (szStrings[i], szStrings[i - 1]) ;
    }
    wcscpy (szStrings[i], pString) ;
    iTotal++ ;

    free (pString) ;
    return TRUE ;
}

EXPORT BOOL CALLBACK DeleteStringA (PCSTR pStringIn)
{
    BOOL    bReturn ;
    int     iLength ;
    PWSTR   pWideStr ;

        // Convert string to Unicode and call DeleteStringW

    iLength = MultiByteToWideChar (CP_ACP, 0, pStringIn, -1, NULL, 0) ;
    pWideStr = malloc (iLength) ;
    MultiByteToWideChar (CP_ACP, 0, pStringIn, -1, pWideStr, iLength) ;
    bReturn = DeleteStringW (pWideStr) ;
```

```
    free (pWideStr) ;

    return bReturn ;
}

EXPORT BOOL CALLBACK DeleteStringW (PCWSTR pStringIn)
{
    int i, j ;

    if (0 == wcslen (pStringIn))
        return FALSE ;

    for (i = 0 ; i < iTotal ; i++)
    {
        if (_wcsicmp (szStrings[i], pStringIn) == 0)
            break ;
    }
        // If given string not in list, return without taking action

    if (i == iTotal)
        return FALSE ;

        // Else adjust list downward

    for (j = i ; j < iTotal ; j++)
        wcscpy (szStrings[j], szStrings[j + 1]) ;

    szStrings[iTotal--][0] = '\0' ;
    return TRUE ;
}

EXPORT int CALLBACK GetStringsA (GETSTRCB pfnGetStrCallBack, PVOID pParam)
{
    BOOL    bReturn ;
    int     i, iLength ;
    PSTR    pAnsiStr ;

    for (i = 0 ; i < iTotal ; i++)
    {
            // Convert string from Unicode

        iLength = WideCharToMultiByte (CP_ACP, 0, szStrings[i], -1, NULL, 0,
                                       NULL, NULL) ;
        pAnsiStr = malloc (iLength) ;
        WideCharToMultiByte (CP_ACP, 0, szStrings[i], -1, pAnsiStr, iLength,
                                       NULL, NULL) ;

            // Call callback function
```

```
            bReturn = pfnGetStrCallBack (pAnsiStr, pParam) ;

            if (bReturn == FALSE)
                return i + 1 ;

            free (pAnsiStr) ;
        }
        return iTotal ;
}

EXPORT int CALLBACK GetStringsW (GETSTRCB pfnGetStrCallBack, PVOID pParam)
{
        BOOL bReturn ;
        int  i ;

        for (i = 0 ; i < iTotal ; i++ )
        {
            bReturn = pfnGetStrCallBack (szStrings[i], pParam) ;
            if (bReturn == FALSE)
                return i + 1 ;
        }
        return iTotal ;
}
```

除了 DllMain 函数以外，STRLIB 中只有 6 个函数供其他函数输出用。所有这些函数都按 EXPORT 定义。这会使 LINK 在 STRLIB.LIB 输入库中列出它们。

21.1.6 STRPROG 程序

STRPROG 程序如程序 21-4 所示，其内容是相当直观的。两个菜单选项(Enter 和 Delete)激活同一个对话框，让您输入一个串，然后 STRPROG 调用 AddString 或 DeleteString。当程序需要更新它的客户区时，请调用 GetStrings，并使用函数 GetStrCallBack 来列出枚举的串。

程序 21-4　STRPROG 程序

STRPROG.C

```
/*-------------------------------------
    STRPROG.C——Program using STRLIB dynamic-link library
                (c) Charles Petzold, 1998
   -------------------------------------*/

#include <windows.h>
#include "strlib.h"
#include "resource.h"
```

```
typedef struct
{
     HDC  hdc ;
     int    xText ;
     int    yText ;
     int    xStart ;
     int    yStart ;
     int    xIncr ;
     int    yIncr ;
     int    xMax ;
     int    yMax ;
}
CBPARAM ;

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
TCHAR szAppName [] = TEXT ("StrProg") ;
TCHAR szString     [MAX_LENGTH + 1] ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = szAppName ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("DLL Demonstration Program"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
```

```
                    CW_USEDEFAULT, CW_USEDEFAULT,
                    NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

BOOL CALLBACK DlgProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
     switch (message)
     {
          case WM_INITDIALOG:
               SendDlgItemMessage (hDlg, IDC_STRING, EM_LIMITTEXT, MAX_LENGTH, 0) ;
               return TRUE ;

          case WM_COMMAND:
               switch (wParam)
               {
               case IDOK:
                    GetDlgItemText (hDlg, IDC_STRING, szString, MAX_LENGTH) ;
                    EndDialog (hDlg, TRUE) ;
                    return TRUE ;

               case IDCANCEL:
                    EndDialog (hDlg, FALSE) ;
                    return TRUE ;
               }
          }
          return FALSE ;
}

BOOL CALLBACK GetStrCallBack (PTSTR pString, CBPARAM * pcbp)
{
     TextOut (pcbp -> hdc, pcbp -> xText, pcbp -> yText,
              pString, lstrlen (pString)) ;

     if ((pcbp -> yText += pcbp -> yIncr) > pcbp -> yMax)
     {
          pcbp -> yText = pcbp -> yStart ;
          if ((pcbp -> xText += pcbp -> xIncr) > pcbp -> xMax)
               return FALSE ;
```

```
    }
    return TRUE ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HINSTANCE  hInst ;
    static int        cxChar, cyChar, cxClient, cyClient ;
    static UINT       iDataChangeMsg ;
    CBPARAM           cbparam ;
    HDC               hdc ;
    PAINTSTRUCT       ps ;
    TEXTMETRIC        tm ;

    switch (message)
    {
    case WM_CREATE:
        hInst = ((LPCREATESTRUCT) lParam) -> hInstance ;
        hdc  = GetDC (hwnd) ;
        GetTextMetrics (hdc, &tm) ;
        cxChar = (int) tm.tmAveCharWidth ;
        cyChar = (int) (tm.tmHeight + tm.tmExternalLeading) ;
        ReleaseDC (hwnd, hdc) ;

            // Register message for notifying instances of data changes

        iDataChangeMsg = RegisterWindowMessage (TEXT ("StrProgDataChange")) ;
        return 0 ;

    case WM_COMMAND:
        switch (wParam)
        {
        case IDM_ENTER:
            if (DialogBox (hInst, TEXT ("EnterDlg"), hwnd, &DlgProc))
            {
                if (AddString (szString))
                    PostMessage (HWND_BROADCAST, iDataChangeMsg, 0, 0) ;
                else
                    MessageBeep (0) ;
            }
            break ;

        case IDM_DELETE:
            if (DialogBox (hInst, TEXT ("DeleteDlg"), hwnd, &DlgProc))
            {
                if (DeleteString (szString))
                    PostMessage (HWND_BROADCAST, iDataChangeMsg, 0, 0) ;
                else
```

```
                    MessageBeep (0) ;
                }
                break ;
            }
            return 0 ;

        case WM_SIZE:
            cxClient = (int) LOWORD (lParam) ;
            cyClient = (int) HIWORD (lParam) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            cbparam.hdc    = hdc ;
            cbparam.xText  = cbparam.xStart = cxChar ;
            cbparam.yText  = cbparam.yStart = cyChar ;
            cbparam.xIncr  = cxChar * MAX_LENGTH ;
            cbparam.yIncr  = cyChar ;
            cbparam.xMax   = cbparam.xIncr * (1 + cxClient / cbparam.xIncr) ;
            cbparam.yMax   = cyChar * (cyClient / cyChar - 1) ;

            GetStrings ((GETSTRCB) GetStrCallBack, (PVOID) &cbparam) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;

        default:
            if (message == iDataChangeMsg)
                InvalidateRect (hwnd, NULL, TRUE) ;
            break ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

STRPROG.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog
```

```
ENTERDLG DIALOG DISCARDABLE 20, 20, 186, 47
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Enter"
FONT 8, "MS Sans Serif"
    BEGIN
    LTEXT           "&Enter:",IDC_STATIC,7,7,26,9
    EDITTEXT        IDC_STRING,31,7,148,12,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,32,26,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,104,26,50,14
END

DELETEDLG DIALOG DISCARDABLE 20, 20, 186, 47
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Delete"
FONT 8, "MS Sans Serif"
BEGIN
    LTEXT           "&Delete:",IDC_STATIC,7,7,26,9
    EDITTEXT        IDC_STRING,31,7,148,12,ES_AUTOHSCROLL
    DEFPUSHBUTTON   "OK",IDOK,32,26,50,14
    PUSHBUTTON      "Cancel",IDCANCEL,104,26,50,14
END

////////////////////////////////////////////////////////////////////////////
// Menu

STRPROG MENU DISCARDABLE
BEGIN
    MENUITEM "&Enter!",             IDM_ENTER
    MENUITEM "&Delete!",            IDM_DELETE
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by StrProg.rc

#define IDC_STRING                  1000
#define IDM_ENTER                   40001
#define IDM_DELETE                  40002
#define IDC_STATIC                  -1
```

STRPROG.C 包含 STRLIB.H 头文件，其中定义了 STRPROG 将使用的 STRLIB 中的三个函数。

当您运行 STRPROG 的多个实例的时候，本程序的奥妙之处就会显露出来。STRLIB 将在共享内存中保存字符串及其指针，并允许 STRPROG 中的所有实例共享此数据。让

我们看一下它是如何运行的吧。

21.1.7 在 STRPROG 实例之间共享数据

Windows 在一个 Win32 进程的地址空间周围筑了一道墙。通常，一个进程的地址空间中的数据是私有的，对别的进程而言是不可见的，但是运行 STRPROG 的多个实例表明 STRLIB 在程序的所有实例之间共享数据是毫无问题的。当您在一个 STRPROG 窗口中添加或删除一个串时，这种改变将立即反映在其他的窗口中。

在它的所有例程之间，STRLIB 共享两个变量：一个字符数组和一个整数（记录已保存的有效串的个数）。STRLIB 将这两个变量保存在作为共享使用的一个特殊内存段中：

```
# pragma data_seg ("shared")
int iTotal = 0 ;
WCHAR szStrings [MAX_STRINGS][MAX_LENGTH + 1] = {'\0'} ;
# pragma data_seg ()
```

第一个 # pragma 语句创建数据段，这里命名为 shared。您可以将这段命名为任何一个您喜欢的名称。在 # pragma 语句之后的所有初始化的变量都进入 shared 段中。第二个 # pragma 语句标志段的结束。对变量进行专门的初始化是很重要的，否则编译程序将把它们放在普通的未初始化的段中而不是放在 shared 中。

链接程序必须知道有一个 shared 段。在 Project Settings 对话框选择 Link 选项卡，选中 STRLIB 时在 Project Options 域（在 Release 和 Debug 配置中均可），包含下面的连接语句：

```
/SECTION:shared,RWS
```

字母 RWS 表明段具有读、写和共享属性。或者，您也可以直接用 DLL 源代码指定连接选项，就像我们在 STRLIB.C 那样：

```
# pragma comment(linker,"/SECTION:shared,RWS")
```

共享的内存段允许 iTotal 变量和 szStrings 字符串数组在 STRLIB 的所有例程之间共享。因为 MAX_STRINGS 字符串数组等于 256，而 MAX_LENGTH 等于 63，所以共享内存段的长度应为32 772字节——iTotal 变量需要 4 字节，256 个指针中的每一个都需要 128 字节。

使用共享内存段可能是在多个应用程序间共享数据的最简单的方法。如果需要动态分配共享内存空间，您应该查看对象映射文件的用法，文档在/Platform SDK/Windows Base Services/Interprocess Communication/File Mapping。

21.2　各种各样的 DLL 主题

如前所述，动态链接库模块不接收消息，但是库模块可以调用 GetMessage 和 PeekMessage。实际上，从消息队列中得到的消息是发给调用库函数的程序的。一般说来，库是代表调用它的程序工作的，这是一条对库所调用的大多数 Windows 函数都适用的规则。

动态链接库可以从库文件或者从调用库的程序文件中加载资源（如图标、串和位图）。加载资源的函数需要实例句柄。如果库使用它自己的实例句柄（初始化期间传给库的），则库能从它自己的文件中获得资源。为了从调用程序的 .EXE 文件中得到资源，库函数需要调用该函数的程序的实例句柄。

在库中登录窗口类和创建窗口需要一点技巧。窗口类结构和 CreateWindow 调用都需要实例句柄。尽管在创建窗口类和窗口时可使用库模块的实例句柄，但在库创建窗口时，窗口消息仍发到调用库的程序的消息队列。如果用户必须在库中创建窗口类和窗口，最好的方法可能是使用调用程序的实例句柄。

因为模态对话框的消息是在程序的消息循环之外接收到的，因此用户可以在库中调用 DialogBox 来创建模态对话框。实例句柄可以是库句柄，并且 DialogBox 的 hwndParent 参数可以为 NULL。

21.2.1　不用输入的动态链接

除了在第一次把用户程序加载到内存中时，由 Windows 执行动态链接外，程序运行时也可以把程序同库模块链接到一起。例如，您通常会这样调用 Rectangle 函数：

```
Rectangle (hdc, xLeft, yTop, xRight, yBottom) ;
```

因为程序和 GDI32.LIB 输入库链接，该库提供了 Rectangle 的地址，因此这种方法有效。

您也可以用更迂回的方法调用 Rectangle。首先用 typedef 为 Rectangle 定义一个函数类型：

```
typedef BOOL (WINAPI * PFNRECT) (HDC, int, int, int, int) ;
```

然后定义两个变量：

```
HANDLE hLibrary ;
PFNRECT pfnRectangle ;
```

现在将 hLibrary 设置为库句柄，将 lpfnRectangle 设置为 Rectangle 函数的地址：

```
hLibrary = LoadLibrary (TEXT ("GDI32.DLL"))
pfnRectangle = (PFNPRECT) GetProcAddress (hLibrary, TEXT ("Rectangle"))
```

如果找不到库文件，或者发生其他一些错误，LoadLibrary 函数返回 NULL。现在您可以

调用如下函数,然后释放库:

```
pfnRectangle (hdc, xLeft, yTop, xRight, yBottom) ;
FreeLibrary (hLibrary) ;
```

尽管这项运行时动态链接的技术并没有为 Rectangle 函数增加多大好处,但它肯定是有用的,如果直到运行时还不知道库模块的名称,这时就需要使用它。

上面的代码使用了 LoadLibrary 和 FreeLibrary 函数。Windows 为所有的库模块提供"引用计数",LoadLibrary 使引用计算递增。当 Windows 加载任何使用库的程序时,引用计数也会递增。FreeLibrary 使引用计数递减,在使用库的程序实例结束时也是如此。当引用计数为 0 时,Windows 将从内存中把库删除掉,因为不再需要它了。

21.2.2　纯资源库

可由 Windows 程序或其他库使用的动态链接库中的任何函数都必须被输出。然而,DLL 也可以不包含任何输出函数。那么,DDL 到底包含什么呢? 答案是资源。

假设用户正在使用需要几幅位图的 Windows 应用程序进行工作。通常要在程序的资源描述文件中列出资源,并用 LoadBitmap 函数把它们加载到内存中。但用户可能希望创建若干套位图,每一套均适用于 Windows 所使用的不同显示适配器。将不同套的位图存放到不同文件中可能是明智的,因为只需要在硬盘上保留一套位图。这些文件就是纯资源文件。

程序 21-5 说明如何创建包含 9 幅位图的名为 BITLIB.DLL 的纯资源库文件。BITLIB.RC 文件列出了所有独立的位图文件,并为每个文件赋予一个序号。为了创建 BITLIB.DLL,需要 9 幅名为 BITMAP1.BMP、BITMAP2.BMP 等位图。您可使用随书附带的光盘上提供的位图,或者在 Visual C++ 中创建这些位图。它们与 ID 号从 1 到 9 相对应。

程序 21-5　BITLIB 库

BITLIB.C

```
/*------------------------------------
   BITLIB.C——Code entry point for BITLIB dynamic-link library
             (c) Charles Petzold, 1998
   ------------------------------------*/

#include <windows.h>

int WINAPI DllMain (HINSTANCE hInstance, DWORD fdwReason, PVOID pvReserved)
{
     return TRUE ;
}
```

BITLIB.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Bitmap

1               BITMAP  DISCARDABLE     "bitmap1.bmp"
2               BITMAP  DISCARDABLE     "bitmap2.bmp"
3               BITMAP  DISCARDABLE     "bitmap3.bmp"
4               BITMAP  DISCARDABLE     "bitmap4.bmp"
5               BITMAP  DISCARDABLE     "bitmap5.bmp"
6               BITMAP  DISCARDABLE     "bitmap6.bmp"
7               BITMAP  DISCARDABLE     "bitmap7.bmp"
8               BITMAP  DISCARDABLE     "bitmap8.bmp"
9               BITMAP  DISCARDABLE     "bitmap9.bmp"
```

在名为 SHOWBIT 的工作空间中创建 BITLIB 项目。在名为 SHOWBIT 的另一个项目中,创建程序 21-6 所示的 SHOWBIT 程序,这与前面的一样。不过,不要使 BITLIB 依靠 SHOWBIT;否则,连接过程中将需要 BITLIB.LIB 文件,并且因为 BITLIB 没有任何输出函数,它也不会创建。事实上,分别连编 BITLIB 和 SHOWBIT,可以交替设置其中一个为"Active Project",然后再连编。

SHOWBIT.C 从 BITLIB 读取位图资源,然后在其客户区显示。按键盘上的任意键可以循环显示。

程序 21-6　SHOWBIT 程序

SHOWBIT.C

```
/*-------------------------------------
   SHOWBIT.C——Shows bitmaps in BITLIB dynamic-link library
               (c) Charles Petzold, 1998
   -------------------------------------*/

# include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("ShowBit") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
```

```
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = (HBRUSH) GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName,
                          TEXT ("Show Bitmaps from BITLIB (Press Key)"),
                          WS_OVERLAPPEDWINDOW,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     if (! hwnd)
          return 0 ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

void DrawBitmap (HDC hdc, int xStart, int yStart, HBITMAP hBitmap)
{
     BITMAP  bm ;
     HDC     hMemDC ;
     POINT   pt ;
```

```
    hMemDC = CreateCompatibleDC (hdc) ;
    SelectObject (hMemDC, hBitmap) ;
    GetObject (hBitmap, sizeof (BITMAP), &bm) ;
    pt.x = bm.bmWidth ;
    pt.y = bm.bmHeight ;

    BitBlt (hdc, xStart, yStart, pt.x, pt.y, hMemDC, 0, 0, SRCCOPY) ;

    DeleteDC (hMemDC) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HINSTANCE  hLibrary ;
    static int        iCurrent = 1 ;
    HBITMAP           hBitmap ;
    HDC               hdc ;
    PAINTSTRUCT       ps ;

    switch (message)
    {
    case WM_CREATE:
        if ((hLibrary = LoadLibrary (TEXT ("BITLIB.DLL"))) == NULL)
        {
            MessageBox (hwnd, TEXT ("Can't load BITLIB.DLL."),
                        szAppName, 0) ;
            return -1 ;
        }
        return 0 ;

    case WM_CHAR:
        if (hLibrary)
        {
            iCurrent ++ ;
            InvalidateRect (hwnd, NULL, TRUE) ;
        }
        return 0 ;

    case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        if (hLibrary)
        {
            hBitmap = LoadBitmap (hLibrary, MAKEINTRESOURCE (iCurrent)) ;

            if (! hBitmap)
            {
                iCurrent = 1 ;
```

```
                hBitmap = LoadBitmap (hLibrary,
                                      MAKEINTRESOURCE (iCurrent)) ;
            }
            if (hBitmap)
            {
                DrawBitmap (hdc, 0, 0, hBitmap) ;
                DeleteObject (hBitmap) ;
            }
        }
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        if (hLibrary)
            FreeLibrary (hLibrary) ;

        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

在处理 WM_CREATE 消息期间，SHOWBIT 获得了 BITLIB.DLL 的句柄：

```
if ((hLibrary = LoadLibrary (TEXT ("BITLIB.DLL"))) == NULL)
```

如果 BITLIB.DLL 与 SHOWBIT.EXE 不在同一个目录，Windows 将按本章前面讨论的方法搜索。如果 LoadLibrary 返回 NULL，SHOWBIT 显示一个消息框来报告错误，并从 WM_CREATE 消息返回 -1。这将导致 WinMain 中的 CreateWindow 调用返回 NULL，而且程序终止。

SHOWBIT 通过库句柄和位图号来调用 LoadBitmap，从而得到一个位图句柄：

```
hBitmap = LoadBitmap (hLibrary, MAKEINTRESOURCE (iCurrent)) ;
```

如果号码 iCurrent 对应的位图无效，或者没有足够的内存来加载位图，则返回一个错误。

在处理 WM_DESTROY 消息时，SHOWBIT 释放库：

```
FreeLibrary (hLibrary) ;
```

当 SHOWBIT 的最后一个实例终止时，BITLIB.DLL 的引用计数变为 0，并且释放占用的内存。这就是实现“剪贴画”程序的一种简单方法，所谓的“剪贴画”程序就是能够将预先创建的位图（或者元文件、增强元文件）加载到供其他程序使用的剪贴板。

第二十二章

声音与音乐

在 Microsoft Windows 中，声音、音乐与视频的综合运用是一个重要的进步。对多媒体的支持起源于 1991 年所谓的 Microsoft Windows 多媒体扩展（Multimedia Extensions to Microsoft Windows）。1992 年，Windows 3.1 的发布使得对多媒体的支持成为另一类 API。最近几年，CD-ROM 驱动器和声卡——在 90 年代初期还很少见——已成为新 PC 的标准配置。现在，几乎所有的人们都深信：多媒体在很大程度上有益于 Windows 的可视化图形，从而使计算机摆脱了只是作为处理数字和文本的机器的传统角色。

22.1 Windows 和多媒体

从某种意义上说，多媒体就是通过与设备无关的函数调用来获得对各种硬件的访问。让我们首先看一下硬件，然后再看看 Windows 多媒体 API 的结构。

22.1.1 多媒体硬件

或许最常用的多媒体硬件就是波形音频设备，也就是平常所说的声卡。波形音频设备将麦克风的输入或其他模拟音频输入转换为数字样本，并将其保存到内存中，或者保存到以 .WAV 为扩展名的磁盘文件中。波形音频设备还将波形转换回模拟声音，以便通过 PC 扬声器来播放。

声卡通常还包含 MIDI 设备。MIDI 是符合工业标准的乐器指令数字化接口（Musical Instrument Digital Interface）。这类硬件播放音符以响应短的二进制消息。MIDI 硬件通常还可以通过电缆连接到如音乐键盘等的 MIDI 输入设备上。通常，外部的 MIDI 合成器也能够添加到声卡上。

现在，大多数 PC 上的 CD-ROM 驱动器都具备播放普通音乐 CD 的能力。这就是平常所说的“CD 音频”。来自波形音频设备、MIDI 设备以及 CD 音频设备的输出，一般在

用户的控制下用“音量控制”程序混合在一起。

另外两个普通的多媒体“设备”不需要额外的硬件。Windows 视频设备(也称为“AVI 视频设备”)播放扩展名为.AVI(audio-video interleave:音频视频插帧)的电影或动画文件。“ActiveMovie 控制”可以播放其他类型的电影,包括 QuickTime 和 MPEG。PC 上的显卡需要特定的硬件来协助播放这些电影。

还有个别 PC 用户使用某种 Pioneer 激光影碟机或者 Sony VISCA 系列盒式磁带录像机。这些设备都有串行接口,因此可由 PC 软件来控制。某些显卡具有一种被称为“窗口影像(video in a window)”的功能,此功能允许一个外部的视频信号与其他应用程序一起出现在 Windows 的屏幕上。这也可以被认为是一种多媒体设备。

22.1.2 API 概述

在 Windows 中,API 支持的多媒体功能主要分成两个集合。它们通常称为“低级”和“高级”接口。

低级接口是一系列函数,这些函数都是以简短的说明性前缀开头,在/Platform SDK/Graphics and Multimedia Services/Multimedia Reference/Multimedia Functions(与高级函数一起)中列出。

低级的波形音频输入输出函数的前缀是 waveIn 和 waveOut。我们将在本章看到这些函数。另外,本章还讨论用 midiOut 函数来控制 MIDI 输出设备。这些 API 还包括 midiIn 和 midiStream 函数。

本章还使用前缀为 time 的函数,这些函数允许设置一个高分辨率的抢占式定时器例程,其定时器的时间间隔速率能够下降到 1 毫秒。此程序主要用于回放 MIDI 乐曲。其他几组函数包括音频压缩、视频压缩以及动画和视频序列,可惜的是本章不包括这些函数。

您还会注意到多媒体函数列表中 7 个带有前缀 mci 的函数,它们允许访问媒体控制接口(MCI:Media Control Interface)。这是一个高级的开放式接口,用于控制多媒体 PC 中所有的多媒体硬件。MCI 包括许多所有多媒体硬件共有的命令。这可能是多媒体的许多方面都以磁带录音机这类设备播放/记录方式为模型的原因。为输入或输出而“打开”一台设备,进而可以“录音”(对于输入)或者“播放”(对于输出),并且结束后可以“关闭”设备。

MCI 本身分为两种形式。一种形式下,可以向 MCI 发送消息,这类似于 Windows 消息。这些消息包括位编码标记和 C 数据结构。另一种形式下,可以向 MCI 发送文本字符串。这个程序主要用于脚本语言,此语言具有灵活的字符串处理函数,但支持调用 Windows API 的函数不多。基于字符串版的 MCI 还有利于交互研究和学习 MCI,我们马上就举一个例子。MCI 中的设备名称包括 cdaudio(CD 音频)、waveaudio(波形声音)、se-

quencer(MIDI 音序器)、videodisc(影碟机)、vcr、overlay(窗口中的模拟视频)、dat(digital audio tape;数字式音频磁带),以及 digitalvideo(数字视频)。MCI 设备分为"简单型"和"混合型"。简单型设备(如 CD 音频)不使用文件;混合型设备(如波形声音)则使用文件。使用波形声音时,这些文件的扩展名是.WAV。

访问多媒体硬件的另一种方法是使用 DirectX API,它超出了本书的范围。

另外两个高级多媒体函数也值得一提:MessageBeep 和 PlaySound,它们在第三章有示范。MessageBeep 播放在"控制面板"的"声音"中指定的声音。PlaySound 可播放磁盘上、内存中,或者作为资源加载的.WAV 文件。本章的后面还会用到 PlaySound 函数。

22.1.3　用 TESTMCI 研究 MCI

在 Windows 多媒体的早期,软件开发工具包含一个名为 MCITEST 的 C 程序,它允许程序员交互地输入 MCI 命令,并学习这些命令的工作方式。这个程序,至少是 C 语言版,显然已经消失了。因此,我又重新创建了它,即程序 22-1 所示的 TESTMCI 程序。这一新创建的程序在用户界面上还是基于老的 MCITEST 程序的,但实际代码不同——尽管我并不认为有什么太大的区别。

程序 22-1　TESTMCI 程序

TESTMCI.C

```
/*------------------------
   TESTMCI.C -- MCI Command String Tester
                (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include "resource.h"

#define ID_TIMER      1

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("TestMci") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     if (-1 == DialogBox (hInstance, szAppName, NULL, DlgProc))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
     }
```

```
    return 0 ;
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static HWND  hwndEdit ;
    int          iCharBeg, iCharEnd, iLineBeg, iLineEnd, iChar, iLine, iLength ;
    MCIERROR     error ;
    RECT         rect ;
    TCHAR        szCommand [1024], szReturn [1024],
                 szError [1024], szBuffer [32] ;

    switch (message)
    {
    case WM_INITDIALOG:
            // Center the window on screen

        GetWindowRect (hwnd, &rect) ;
        SetWindowPos (hwnd, NULL,
          (GetSystemMetrics (SM_CXSCREEN) - rect.right + rect.left) / 2,
          (GetSystemMetrics (SM_CYSCREEN) - rect.bottom + rect.top) / 2,
          0, 0, SWP_NOZORDER | SWP_NOSIZE) ;

        hwndEdit = GetDlgItem (hwnd, IDC_MAIN_EDIT) ;
        SetFocus (hwndEdit) ;
        return FALSE ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDOK:
                // Find the line numbers corresponding to the selection
            SendMessage (hwndEdit, EM_GETSEL, (WPARAM) &iCharBeg,
                                              (LPARAM) &iCharEnd) ;

            iLineBeg = SendMessage (hwndEdit, EM_LINEFROMCHAR, iCharBeg, 0) ;
            iLineEnd = SendMessage (hwndEdit, EM_LINEFROMCHAR, iCharEnd, 0) ;

                // Loop through all the lines

            for (iLine = iLineBeg ; iLine <= iLineEnd ; iLine++)
            {
                    // Get the line and terminate it; ignore if blank

                * (WORD *) szCommand = sizeof (szCommand) / sizeof (TCHAR) ;

                iLength = SendMessage (hwndEdit, EM_GETLINE, iLine,
```

```
                                        (LPARAM) szCommand) ;
            szCommand [iLength] = '\0' ;

            if (iLength == 0)
                continue ;

                // Send the MCI command

            error = mciSendString (szCommand, szReturn,
                                   sizeof (szReturn) / sizeof (TCHAR), hwnd) ;

                // Set the Return String field

            SetDlgItemText (hwnd, IDC_RETURN_STRING, szReturn) ;

                // Set the Error String field (even if no error)

            mciGetErrorString (error, szError,
                               sizeof (szError) / sizeof (TCHAR)) ;

            SetDlgItemText (hwnd, IDC_ERROR_STRING, szError) ;
        }
            // Send the caret to the end of the last selected line

        iChar = SendMessage (hwndEdit, EM_LINEINDEX, iLineEnd, 0) ;
        iChar += SendMessage (hwndEdit, EM_LINELENGTH, iCharEnd, 0) ;
        SendMessage (hwndEdit, EM_SETSEL, iChar, iChar) ;

            // Insert a carriage return/line feed combination

        SendMessage (hwndEdit, EM_REPLACESEL, FALSE,
                               (LPARAM) TEXT ("\r\n")) ;
        SetFocus (hwndEdit) ;
        return TRUE ;

    case IDCANCEL:
        EndDialog (hwnd, 0) ;
        return TRUE ;

    case IDC_MAIN_EDIT:
        if (HIWORD (wParam) == EN_ERRSPACE)
        {
            MessageBox (hwnd, TEXT ("Error control out of space."),
                        szAppName, MB_OK | MB_ICONINFORMATION) ;
            return TRUE ;
        }
        break ;
```

```
            }
            break ;

        case MM_MCINOTIFY:
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_MESSAGE), TRUE) ;

            wsprintf (szBuffer, TEXT ("Device ID = %i"), lParam) ;
            SetDlgItemText (hwnd, IDC_NOTIFY_ID, szBuffer) ;
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_ID), TRUE) ;

            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_SUCCESSFUL),
                          wParam & MCI_NOTIFY_SUCCESSFUL) ;

            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_SUPERSEDED),
                          wParam & MCI_NOTIFY_SUPERSEDED) ;

            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_ABORTED),
                          wParam & MCI_NOTIFY_ABORTED) ;

            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_FAILURE),
                          wParam & MCI_NOTIFY_FAILURE) ;

            SetTimer (hwnd, ID_TIMER, 5000, NULL) ;
            return TRUE ;
        case WM_TIMER:
            KillTimer (hwnd, ID_TIMER) ;

            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_MESSAGE), FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_ID), FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_SUCCESSFUL), FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_SUPERSEDED), FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_ABORTED), FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_NOTIFY_FAILURE), FALSE) ;
            return TRUE ;

        case WM_SYSCOMMAND:
            switch (LOWORD (wParam))
            {
            case SC_CLOSE:
                EndDialog (hwnd, 0) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}
```

TESTMCI.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog

TESTMCI DIALOG DISCARDABLE  0, 0, 270, 276
STYLE WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "MCI Tester"
FONT 8, "MS Sans Serif"
BEGIN
    EDITTEXT        IDC_MAIN_EDIT,8,8,254,100,ES_MULTILINE | ES_AUTOHSCROLL |
                    WS_VSCROLL
    LTEXT           "Return String:",IDC_STATIC,8,114,60,8
    EDITTEXT        IDC_RETURN_STRING,8,126,120,50,ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | WS_GROUP | NOT WS_TABSTOP
    LTEXT           "Error String:",IDC_STATIC,142,114,60,8
    EDITTEXT        IDC_ERROR_STRING,142,126,120,50,ES_MULTILINE |
                    ES_AUTOVSCROLL | ES_READONLY | NOT WS_TABSTOP
    GROUPBOX        "MM_MCINOTIFY Message",IDC_STATIC,9,186,254,58
    LTEXT           "",IDC_NOTIFY_ID,26,198,100,8
    LTEXT           "MCI_NOTIFY_SUCCESSFUL",IDC_NOTIFY_SUCCESSFUL,26,212,100,
                    8,WS_DISABLED
    LTEXT           "MCI_NOTIFY_SUPERSEDED",IDC_NOTIFY_SUPERSEDED,26,226,100,
                    8,WS_DISABLED
    LTEXT           "MCI_NOTIFY_ABORTED",IDC_NOTIFY_ABORTED,144,212,100,8,
                    WS_DISABLED
    LTEXT           "MCI_NOTIFY_FAILURE",IDC_NOTIFY_FAILURE,144,226,100,8,
                    WS_DISABLED
    DEFPUSHBUTTON   "OK",IDOK,57,255,50,14
    PUSHBUTTON      "Close",IDCANCEL,162,255,50,14
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by TestMci.rc

#define IDC_MAIN_EDIT                   1000
#define IDC_NOTIFY_MESSAGE              1005
#define IDC_NOTIFY_ID                   1006
#define IDC_NOTIFY_SUCCESSFUL           1007
```

```
# define IDC_NOTIFY_SUPERSEDED        1008
# define IDC_NOTIFY_ABORTED           1009
# define IDC_NOTIFY_FAILURE           1010
# define IDC_SIGNAL_MESSAGE           1011
# define IDC_SIGNAL_ID                1012
# define IDC_SIGNAL_PARAM             1013
# define IDC_RETURN_STRING            1014
# define IDC_ERROR_STRING             1015
# define IDC_DEVICES                  1016
# define IDC_STATIC                   -1
```

与本章的大多数程序一样，TESTMCI 使用非模态对话框作为它的主窗口。与本章所有的程序一样，TESTMCI 要求 WINMM.LIB 导入库在 Microsoft Visual C++ 的 Projects Settings 对话框的 Links 页列出。

此程序用到了两个最重要的多媒体函数：mciSendString 和 mciGetErrorText。在 TESTMCI 的主编辑窗口输入一些内容并按下 Enter 键（或 OK 按钮）后，程序将输入的字符串作为第一个参数传递给 mciSendString 命令：

```
error = mciSendString (szCommand, szReturn,
                       sizeof (szReturn) / sizeof (TCHAR), hwnd) ;
```

如果在编辑窗口选择了不止一行，则程序将按顺序把它们发送给 mciSendString 函数。第二个参数是字符串地址，此字符串获取从函数返回的信息。程序将此信息显示在窗口的 Return String 区域。从 mciSendString 返回的错误代码传递给 mciGetErrorString 函数，以获得文本出错说明，此说明显示在 TESTMCI 窗口的 Error String 区域。

22.1.4　MCITEXT 和 CD 音频

通过控制 CD-ROM 驱动器和播放音频 CD，您可以对 MCI 命令字符串留下很好的印象。因为这些命令字符串一般都非常简单，并且更重要的是您可以听到一些音乐，所以这是好的起点。您还可以在/Platform SDK/Graphics and Multimedia Services/Multimedia Reference/Multimedia Command Strings 中获得 MCI 命令字符串的参考，以方便本练习。

请确认 CD-ROM 驱动器的音频输出已连接到扬声器或耳机，然后放入一张音频 CD，如 Bruce Springsteen 的《Born to Run》。在 Windows 98 中，“CD 播放器”将启动并开始播放此唱盘。如果是这样的话，终止“CD 播放器”。作为替代，可以调出 TESTMCI，并且键入命令

```
open cdaudio
```

然后按 Enter 键。其中 open 是 MCI 命令，cdaudio 是 MCI 认定的 CD-ROM 驱动器的设备名称。（假定您的系统中只有一个 CD-ROM 驱动器。要获得多个 CD-ROM 驱动器名称

需使用 sysinfo 命令。)

TESTMCI 中的 Return String 区域显示 mciSendString 函数中系统返回给程序的字符串。如果运行 open 命令,则此值是 1。TESTMCI 在 Error String 区域中显示 mciGetErrorString 基于 mciSendString 返回值所返回的信息。如果 mciSendString 没有返回报错代码,则 Error String 区域显示文本 The specified command was carried out。

假定执行了 open 命令,现在就可以输入

```
play cdaudio
```

CD 将开始播放唱盘上的第一首乐曲《Thunder Road》。输入命令

```
pause cdaudio
```

或者

```
stop cdaudio
```

可以暂停播放。

对于 CD 音频设备来说,这些语句的功能相同。您可用下面的语句重新播放:

```
play cdaudio
```

迄今为止,我们使用的全部字符串都由命令和设备名称组成,其中有些命令带有选项。例如,键入

```
status cdaudio position
```

根据收听时间的长短,Return String 区域将显示类似下面的一些字符:

```
01:15:25
```

这是些什么?很显然不是小时、分和秒,因为 CD 没有那么长。要找出时间格式,请键入

```
status cdaudio time format
```

现在 Return String 区域显示下面的字符串:

```
msf
```

这代表"分-秒-帧"。CD 音频中,每秒有 75 帧。时间格式的帧部分可在 0 到 74 之间的范围内变化。

状态命令有一连串的选项。使用下面的命令,您可以确定 msf 格式的 CD 全部长度:

```
status cdaudio length
```

对于《Born to Run》,Return String 区域将显示:

```
39:28:19
```

这指的是 39 分 28 秒 19 帧。

现在试一下

```
status cdaudio number of tracks
```

Return String 区域将显示：

```
8
```

我们从 CD 封面上知道《Born to Run》CD 上第 5 首乐曲是主题曲。MCI 命令中的乐曲从 1 开始编号。要想知道乐曲“Born to Run”的长度，可以键入下面的命令：

```
status cdaudio length track 5
```

“Return String”区域将显示：

```
04:30:22
```

我们还可确定此乐曲从盘上的哪个位置开始：

```
status cdaudio position track 5
```

“Return String”区域将显示：

```
17:36:35
```

根据这条信息，我们可以直接跳到乐曲标题：

```
play cdaudio from 17:36:35 to 22:06:57
```

此命令只播放一首乐曲，然后停止。最后的值是由 4:30:22(乐曲长度)加 17:36:35 得到的。或者，也可以用下面的命令确定：

```
status cdaudio position track 6
```

或者，也可以将时间格式设置为乐曲-分-秒-帧：

```
set cdaudio time format tmsf
```

然后具体地设为：

```
play cdaudio from 5:0:0:0 to 6:0:0:0
```

或者，更简单地设为：

```
play cdaudio from 5 to 6
```

如果时间的尾部是 0，那么您可去掉它们。还可以用毫秒设置时间格式。

每个 MCI 命令字符串都可以在字符串的后面包括选项 wait 和 notify(但不是同时使

用)。例如,假设您只想播放《Born to Run》的前 10 秒,而且播放后,您还想让程序完成其他工作,可以按下面的方法进行(假定您已经将时间格式设置为 tmsf):

```
play cdaudio from 5:0:0 to 5:0:10 wait
```

这种情况下,直到函数运行结束,也就是说,直到播放完《Born to Run》的前 10 秒,mciSendString 函数才返回。

现在很明显,一般来说,在单线程的应用程序中这不是一件好事。如果偶然键入

```
play cdaudio wait
```

直到整个唱盘播放完以后,mciSendString 函数才将控制返回给程序。如果必须使用 wait 选项(在运行 MCI 描述文件的时候,它很方便,与我将演示的一样),首先使用 break 命令。此命令可设置一个虚拟键码,此码将中断 mciSendString 命令,并将控制返回给程序。例如,要设置 Escape 键来实现此目的,可用

```
break cdaudio on 27
```

这里,27 是十进制的 VK_ESCAPE 值。

比 wait 选项更好的是 notify 选项:

```
play cdaudio from 5:0:0 to 5:0:10 notify
```

这种情况下,mciSendString 函数立即返回,但如果该操作在 MCI 命令的尾部定义,那么其句柄被指定为 mciSendString 函数最后一个参数的窗口会接收到 MM_MCINOTIFY 消息。TESTMCI 程序在 MM_MCINOTIFY 框中显示此消息的结果。为避免与其他可能键入的命令混淆,TESTMCI 程序在 5 秒后停止显示 MM_MCINOTIFY 消息的结果。

您可以同时使用 wait 和 notify 关键字,但没有理由这么做。不使用这两个关键字,默认的操作就既不是 wait,也不是您通常所希望的 notify。

用这些命令结束播放时,可键入下面的命令来停止 CD:

```
stop cdaudio
```

如果在关闭之前没有停止 CD-ROM 设备,那么甚至在关闭设备之后还会继续播放 CD。

另外,您还可以试试您的硬件是否支持一些特殊的命令:

```
eject cdaudio
```

最后按下面的方法关闭设备:

```
close cdaudio
```

虽然 TESTMCI 自己不能保存或加载文本文件,但可以在编辑控制和剪贴板之间复制文本:先从 TESTMCI 选择一些内容,将其复制到剪贴板(用 Ctrl-C),再将这些文本从

剪贴板复制到“记事本”，然后保存。相反的操作，可以将一系列的 MCI 命令加载到 TESTMCI。如果选择了一系列命令，然后按下 OK 按钮（或者 Enter 键），那么 TESTMCI 将每次执行一条命令。这就允许您编写 MCI 的“描述文件”，也就是 MCI 命令的简单列表。

例如，假设您想听歌曲《Jungleland》（盘中的最后一首）、《Thunder Road》和《Born to Run》，并要按此顺序听，可以编写如下的脚本：

```
open cdaudio
set cdaudio time format tmsf
break cdaudio on 27
play cdaudio from 8 wait
play cdaudio from 1 to 2 wait
play cdaudio from 5 to 6 wait
stop cdaudio
eject cdaudio
close cdaudio
```

不用 wait 关键字，就不能正常工作，因为 mciSendString 命令会立即返回，然后执行下一条命令。

此时，编写模拟 CD 播放器的简单的应用程序，就应该相当清楚了。程序可以确定乐曲数量、每个乐曲的长度，并能显示允许用户从任意位置开始播放。（不过，请记住：mciSendString 总是返回文本字符串信息，因此您需要编写解析逻辑来将这些字符串转换成数字。）可以肯定，这样的程序还要使用 Windows 计时器，以产生大约 1 秒的时间间隔。在 WM_TIMER 消息期间，程序将调用

```
status cdaudio mode
```

来查看 CD 是暂停还是在播放。命令

```
status cdaudio position
```

允许程序更新显示，以为用户显示当前的位置。但可能还存在更令人感兴趣的事：如果程序知道音乐音调部分的节拍位置，那么就可以使屏幕上的图形与 CD 同步。这对于音乐指令或者创建自己的图形音乐视频程序极为有用。

22.2 波形音频

波形音频是最常用的 Windows 多媒体特性。波形音频设备可以通过麦克风捕获声音，并将其转换为数字，然后把它们保存到内存或者磁盘上的波形文件中，波形文件的扩展名是.WAV。这样，声音就可以播放了。

22.2.1 声音和波形

在接触波形音频 API 之前，具备一些预备知识很重要，这些知识包括物理学、听觉，以及声音进出计算机的过程。

声音就是振动。当声波传入人耳改变了耳膜上空气的压力时，我们就会感觉到声音。麦克风可以拾取这些振动，并且将它们转换为电流。同样，电流再经过放大器和扬声器，就又变成了声音。传统上，声音按模拟方式存储（例如录音磁带和唱片），这些振动存储在磁脉冲或者轮廓凹槽中。当声音转换为电流时，就可以用随时间振动的波形来表示。振动最自然的形式可以用正弦波表示，它的一个周期如上册中第五章图 5-5 所示。

正弦波有两个参数——振幅（也就是一个周期中的最大振幅）和频率。我们通过音量认识振幅，通过音调认识频率。一般来说，人耳可感受的正弦波的范围是从20 Hz（每秒周期）的低频声到 20 000 Hz 的高频声，但随着年龄的增长，对高频声的感受能力会逐年减弱。

人感受频率的能力，要求该频率是对数关系而不是线性关系，也就是说，我们感受 20 ~ 40 Hz的频率变化与感受 40 ~ 80 Hz 的频率变化是一样的。在音乐中，这种加倍的频率定义为八度音阶。因此，人耳可感觉到大约 10 个八度音阶的声音。钢琴的范围是从 27.5 ~ 4186 Hz 之间，略小于 7 个八度音阶。

虽然正弦波代表了振动的大多数自然形式，但纯正弦波很少在现实生活中单独出现，而且，纯正弦波并不动听。大多数声音都很复杂。

任何周期的波形（即一个循环波形）可以分解成多个正弦波，这些正弦波的频率都是整倍数。这就是所谓的 Fourier 级数，它以法国数学家和物理学家 Jean Baptiste Joseph Fourier(1768 ~ 1830)的名字命名。周期的频率是基础。级数中其他正弦波的频率是基础频率的 2 倍、3 倍、4 倍等等。这些频率的声音称为泛音。基础频率也称作一级谐波。第一泛音是二级谐波，以此类推。

正弦波谐波的相对强度给每个周期的波形唯一的声音，这就是“音质”，它使得喇叭吹出喇叭声，钢琴弹出钢琴声。

人们一度认为电子合成乐器仅仅需要将声音分解成谐波，并与多个正弦波重组即可。不过，事实证明现实世界中的声音并不是这么简单。代表现实世界中声音的波形都没有严格的周期。乐器之间谐波的相对强度是不同的，并且谐波也随着每个音符的演奏时间改变，特别是乐器演奏音符的开始位置——我们称为“冲击（attact）”——相当复杂，但这个位置又对我们感受音质至关重要。

由于近年来数字存储能力的提高，我们可以将声音直接以数字形式存储，而不用复杂的重组。

22.2.2 脉冲编码调制

计算机处理的是数字,因此要使声音进入计算机,就必须设计一种能将声音与数字信号相互转换的机制。

不压缩数据就完成此功能的最常用的方法称作“脉冲编码调制(PCM:pulse code modulation)”。PCM 可用于光盘、数字式录音磁带以及 Windows 中。脉冲编码调制是在概念上简化的处理过程的奇怪术语。

利用脉冲编码调制,波形可以按固定的周期频率采样,其频率通常是每秒几万次。对于每个样本都测量其波形的振幅。完成将振幅转换成数字工作的硬件是模/数转换器(ADC:analog-to-digital converter)。类似地,通过数/模转换器(DAC:digital-to-analog converter)可将数字转换回波形电信号,但这样转换得到的波形与输入的并不完全相同。合成的波形具有由高频组成的尖锐边缘。因此,播放硬件通常在数/模转换器后还包括一个低通滤波器。此滤波器滤掉高频,并使合成后的波形更平滑。在输入端,低通滤波器位于 ADC 前面。

脉冲编码调制有两个参数:采样速率,即每秒内测量波形振幅的次数;样本大小,即用于存储振幅级的位数。与您想象的一样:采样速率越高,样本大小越大,原始声音的复制品才更好。不过,存在一个提高采样速率和样本大小的极点,超过这个极点也就超过了人类分辨声音的极限。另外,如果采样速率和样本大小过低,将导致不能精确地复制音乐及其他声音。

22.2.3 采样速率

采样速率决定声音可被数字化和存储的最大频率。尤其是,采样速率必须是样本声音最高频率的两倍,这就是“Nyquist 频率(Nyquist Frequency)”,以 30 年代研究采样过程的工程师 Harry Nyquist 的名字命名。

以过低的采样速率对正弦波采样时,合成的波形比最初的波形频率更低,这就是所说的伪信号。为避免伪信号的发生,在输入端使用低通滤波器以阻止频率大于半个采样速率的所有波形。在输出端,数/模转换器产生的粗糙的波形边缘实际上是由频率大于半个采样速率的波形组成的泛音。因此,位于输出端的低通滤波器也阻止频率大于半个采样速率的所有波形。

音频 CD 中使用的采样速率是每秒 44 100 个样本,或者称为44.1 kHz。这个特有的数字是这样产生的:

人耳可听到最高 20 kHz 的声音,因此要捕获人能听到的整个音频范围,就需要 40 kHz的采样速率。然而,由于低通滤波器具有跌落效果,所以采样速率应该再高出大约 10%才行。现在,采样速率就达到了44 kHz。这时,我们要与视频同时记录数字音

频，于是采样速率就应该是美国、欧洲电视帧速率的整数倍，这两种帧速率分别是30 Hz和25 Hz。这就使采样速率升高到了44.1 kHz。

采样速率为44.1 kHz的光盘会产生大量的数据，这对于一些应用程序来说实在是太多了，例如对于录制声音而不是录制音乐时就是这样。把采样速率减半到22.05 kHz，可由一个10 kHz的泛音来简化复制声音的上半部分，再将其减半到11.025 kHz就向我们提供了5 kHz频率范围。44.1 kHz、22.05 kHz和11.025 kHz的采样速率，以及8 kHz都是波形音频设备普遍支持的标准。

因为钢琴的最高频率为4186 Hz，所以您可能会认为给钢琴录音时，11.025 kHz的采样速率就足够了。但钢琴最高的基础频率是4186 Hz。滤掉大于5000 Hz的所有正弦波将减少可被复制的泛音，而这样将不能精确地捕获和复制钢琴的声音。

22.2.4 样本大小

脉冲编码调制的第二个参数是按位计算的样本大小。样本大小决定了可供录制和播放的最低音与最高音之间的区别。这就是通常所说的动态范围。

声音强度是波形振幅的平方（即每个正弦波一个周期中最大振幅的合成）。与频率一样，人对声音强度的感受也呈对数变化。

两个声音在强度上的差是以贝尔（以电话发明人 Alexander Graham Bell 的名字命名的）和分贝（dB）为单位来度量的。两个声音的强度相差10倍时，即说两者声强差1贝尔。1分贝等于0.1贝尔。经计算可知，1分贝在声音强度上是以1.26倍（10的10次方根）增加的，在波形振幅上是以1.12倍（10的20次方根）增加的。1分贝是人耳可辨识的最小声音强度增量。从刚刚能听到声音极限到让人无法忍受的声音极限间的声音强度差约为100分贝。

您可以通过下面的算式以分贝为单位来计算两个声音间的动态范围：

$$20\cdot\log\left(\frac{A_1}{A_2}\right)\text{分贝}$$

其中 A_1 和 A_2 是两个声音的振幅。因为只可能有一个振幅，所以样本大小为1位，其动态范围为0。

若样本大小为8位，则最大振幅与最小振幅的比值为256。这样，动态范围就是：

$$20\cdot\log(256)\text{分贝}$$

即48分贝。48分贝的动态范围大约相当于安静的房屋与电动割草机之间的差别。若将样本大小加倍到16位，则产生的动态范围是：

$$20\cdot\log(65536)\text{分贝}$$

即96分贝。这非常接近于听觉极限与痛苦极限的差别，并且它被认为是再现音乐的理想范围。

Windows 同时支持 8 位和 16 位的样本大小。保存 8 位的样本时，样本按无符号字节处理。静音将保存为一个值为 0x80 的字符串；16 位的样本按符号整数处理，这时静音将保存为一个值为 0 的字符串。

要计算未压缩音频所需的存储空间，可用以秒为单位的声音持续时间乘以采样速率。如果用 16 位样本而不是 8 位样本，则将其加倍；如果是录制立体声，则再加倍。例如，1 小时的 CD 声音（或者是在每个立体声样本占 2 字节、每秒 44 100 个样本的速度下进行 3600 秒）需要635 MB，这十分接近一张 CD-ROM 的存储量。

22.2.5　在软件中产生正弦波

对于第一个关于波形音频的练习，我们不打算将声音保存到文件中或播放录制的声音。我们将使用低级的波形音频 API（前缀是 waveOut 的函数）来创建一个称作 SINEWAVE 的音频正弦波生成器。此程序按 1 Hz 的增量来生成从 20 Hz（人可感觉的最低值）到 5000 Hz（与人感觉的最高值相差两个八度音阶）的正弦波。

我们知道，标准 C 运行库包括一个 sin 函数，该函数返回一个弧度角的正弦值。（2π 弧度等于 360°。）sin 函数返回值的范围是 -1 ~ 1。（早在第五章，我们就在 SINEWAVE 程序中使用过这个函数。）因此，应该很容易使用 sin 函数生成输出到波形音频硬件的正弦波数据。基本上是用代表波形（这时是正弦波）的数据来填充缓冲区，并将此缓冲区传递给 API。（这比前面所讲的稍微有些复杂，但我将详细介绍。）波形音频硬件播放完缓冲区中的数据后，应将第二个缓冲区中的数据传递给它，并且以此类推。

第一次考虑这个问题（而且对 PCM 也一无所知）时，您大概会认为将一个周期的正弦波分成若干固定数量的样本——例如 360 个——才合理。对于 20 Hz 的正弦波，每秒输出 7200 个样本。对于 200 Hz 的正弦波，每秒则要输出 72 000 个样本。这有可能实现，但实际上却不能这么做。对于 5000 Hz 的正弦波，就需要每秒输出 180 万个样本，这的确会增大 DAC 的负担！更重要的是，对于更高的频率，这种做法会比实际需要的精度还高。

就脉冲编码调制而言，采样速率是个常量。假定采样速率是 SINEWAVE 程序中使用的11 025 Hz。如果要生成一个2756.25 Hz（确切地说是 1/4 的采样速率）的正弦波，则正弦波的每个周期就有 4 个样本。对于 25 Hz 的正弦波，每个周期就有 441 个样本。通常，每周期的样本数等于采样速率除以要得到的正弦波频率。一旦知道了每周期的样本数，用 2π 弧度除以此数，然后用 sin 函数来获得每周期的样本，然后再反复对一个周期进行采样，从而创建一个连续的波形。

问题是每周期的样本数可能带有小数，因此在使用时这种方法并不是很好。每个周期的尾部都会有间断。

使它正常工作的关键是保留一个静态的“相位角”变量。此角初始化为 0。第一个

样本是 0° 正弦。随后，相位角增加一个值，该值等于 2π 乘以频率再除以采样速率。用此相位角作为第二个样本，并且按此方法继续。一旦相位角超过 2π 弧度，则减去 2π 弧度，而不要把相位角再初始化为 0。

例如，假定要用 11 025 Hz 的采样速率来生成 1000 Hz 的正弦波。即每周期有大约 11 个样本。为便于理解，此处相位角按度数给出——大约前一个半周期的相位角是：0、32.65、65.31、97.96、130.61、163.27、195.92、228.57、261.22、293.88、326.53、359.18、31.84、64.49、97.14、129.80、162.45、195.10，以此类推。存入缓冲区的波形数据是这些角度的正弦值，并已缩放到每样本的位数。为后来的缓冲区创建数据时，可继续增加最后的相位角，而不要将它初始化为 0。

如程序 22-2 所示，FillBuffer 函数完成这项工作——与 SINEWAVE 程序的其余部分一起完成。

程序 22-2　SINEWAVE 程序

SINEWAVE.C

```
/*------------------------------------------
   SINEWAVE.C——Multimedia Windows Sine Wave Generator
                (c) Charles Petzold, 1998
   ------------------------------------------*/

#include <windows.h>
#include <math.h>
#include "resource.h"

#define SAMPLE_RATE             11025
#define FREQ_MIN                20
#define FREQ_MAX                5000
#define FREQ_INIT               440
#define OUT_BUFFER_SIZE         4096
#define PI                      3.14159

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("SineWave") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     if (-1 == DialogBox (hInstance, szAppName, NULL, DlgProc))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
```

```
    }
    return 0 ;
}

VOID FillBuffer (PBYTE pBuffer, int iFreq)
{
    static double fAngle ;
    int          i ;

    for (i = 0 ; i < OUT_BUFFER_SIZE ; i++ )
    {
        pBuffer [i] = (BYTE) (127 + 127 * sin (fAngle)) ;

        fAngle += 2 * PI * iFreq / SAMPLE_RATE ;

        if (fAngle > 2 * PI)

            fAngle -= 2 * PI ;
    }
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BOOL          bShutOff, bClosing ;
    static HWAVEOUT      hWaveOut ;
    static HWND          hwndScroll ;
    static int           iFreq = FREQ_INIT ;
    static PBYTE         pBuffer1, pBuffer2 ;
    static PWAVEHDR      pWaveHdr1, pWaveHdr2 ;
    static WAVEFORMATEX  waveformat ;
    int                  iDummy ;

    switch (message)
    {
    case WM_INITDIALOG:
        hwndScroll = GetDlgItem (hwnd, IDC_SCROLL) ;
        SetScrollRange (hwndScroll, SB_CTL, FREQ_MIN, FREQ_MAX, FALSE) ;
        SetScrollPos   (hwndScroll, SB_CTL, FREQ_INIT, TRUE) ;
        SetDlgItemInt  (hwnd, IDC_TEXT, FREQ_INIT, FALSE) ;

        return TRUE ;

    case WM_HSCROLL:
        switch (LOWORD (wParam))
        {
        case SB_LINELEFT:   iFreq -= 1 ;   break ;
        case SB_LINERIGHT:  iFreq += 1 ;   break ;
```

```
        case SB_PAGELEFT:  iFreq /= 2 ;  break ;
        case SB_PAGERIGHT: iFreq *= 2 ;  break ;

        case SB_THUMBTRACK:
            iFreq = HIWORD (wParam) ;
            break ;

        case SB_TOP:
            GetScrollRange (hwndScroll, SB_CTL, &iFreq, &iDummy) ;
            break ;

        case SB_BOTTOM:
            GetScrollRange (hwndScroll, SB_CTL, &iDummy, &iFreq) ;
            break ;
        }

        iFreq = max (FREQ_MIN, min (FREQ_MAX, iFreq)) ;
        SetScrollPos (hwndScroll, SB_CTL, iFreq, TRUE) ;
        SetDlgItemInt (hwnd, IDC_TEXT, iFreq, FALSE) ;
        return TRUE ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDC_ONOFF:
                // If turning on waveform, hWaveOut is NULL

            if (hWaveOut == NULL)
            {
                    // Allocate memory for 2 headers and 2 buffers

                pWaveHdr1 = malloc (sizeof (WAVEHDR)) ;
                pWaveHdr2 = malloc (sizeof (WAVEHDR)) ;
                pBuffer1 = malloc (OUT_BUFFER_SIZE) ;
                pBuffer2 = malloc (OUT_BUFFER_SIZE) ;

                if (!pWaveHdr1 || !pWaveHdr2 || !pBuffer1 || !pBuffer2)
                {
                    if (!pWaveHdr1) free (pWaveHdr1) ;
                    if (!pWaveHdr2) free (pWaveHdr2) ;
                    if (!pBuffer1) free (pBuffer1) ;
                    if (!pBuffer2) free (pBuffer2) ;

                    MessageBeep (MB_ICONEXCLAMATION) ;
                    MessageBox (hwnd, TEXT ("Error allocating memory!"),
                                szAppName, MB_ICONEXCLAMATION | MB_OK) ;
```

```
        return TRUE ;
    }
        // Variable to indicate Off button pressed

    bShutOff = FALSE ;

        // Open waveform audio for output

    waveformat.wFormatTag          = WAVE_FORMAT_PCM ;
    waveformat.nChannels           = 1 ;
    waveformat.nSamplesPerSec      = SAMPLE_RATE ;
    waveformat.nAvgBytesPerSec     = SAMPLE_RATE ;
    waveformat.nBlockAlign         = 1 ;
    waveformat.wBitsPerSample      = 8 ;
    waveformat.cbSize              = 0 ;

    if (waveOutOpen (&hWaveOut, WAVE_MAPPER, &waveformat,
                (DWORD) hwnd, 0, CALLBACK_WINDOW)
            != MMSYSERR_NOERROR)
    {
        free (pWaveHdr1) ;
        free (pWaveHdr2) ;
        free (pBuffer1) ;
        free (pBuffer2) ;

        hWaveOut = NULL ;
        MessageBeep (MB_ICONEXCLAMATION) ;
        MessageBox (hwnd,
            TEXT ("Error opening waveform audio device!"),
            szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return TRUE ;
    }

        // Set up headers and prepare them

    pWaveHdr1 -> lpData            = pBuffer1 ;
    pWaveHdr1 -> dwBufferLength    = OUT_BUFFER_SIZE ;
    pWaveHdr1 -> dwBytesRecorded   = 0 ;
    pWaveHdr1 -> dwUser            = 0 ;
    pWaveHdr1 -> dwFlags           = 0 ;
    pWaveHdr1 -> dwLoops           = 1 ;
    pWaveHdr1 -> lpNext            = NULL ;
    pWaveHdr1 -> reserved          = 0 ;

    waveOutPrepareHeader (hWaveOut, pWaveHdr1,
                          sizeof (WAVEHDR)) ;
    pWaveHdr2 -> lpData            = pBuffer2 ;
```

```
                pWaveHdr2 -> dwBufferLength      = OUT_BUFFER_SIZE ;
                pWaveHdr2 -> dwBytesRecorded     = 0 ;
                pWaveHdr2 -> dwUser              = 0 ;
                pWaveHdr2 -> dwFlags             = 0 ;
                pWaveHdr2 -> dwLoops             = 1 ;
                pWaveHdr2 -> lpNext              = NULL ;
                pWaveHdr2 -> reserved            = 0 ;

                waveOutPrepareHeader (hWaveOut, pWaveHdr2,
                                      sizeof (WAVEHDR)) ;
            }
            // If turning off waveform, reset waveform audio
            else
            {
                bShutOff = TRUE ;
                waveOutReset (hWaveOut) ;
            }
            return TRUE ;
        }
        break ;

        // Message generated from waveOutOpen call

    case MM_WOM_OPEN:
        SetDlgItemText (hwnd, IDC_ONOFF, TEXT ("Turn Off")) ;

        // Send two buffers to waveform output device

        FillBuffer (pBuffer1, iFreq) ;
        waveOutWrite (hWaveOut, pWaveHdr1, sizeof (WAVEHDR)) ;

        FillBuffer (pBuffer2, iFreq) ;
        waveOutWrite (hWaveOut, pWaveHdr2, sizeof (WAVEHDR)) ;
        return TRUE ;

        // Message generated when a buffer is finished

    case MM_WOM_DONE:
        if (bShutOff)
        {
            waveOutClose (hWaveOut) ;
            return TRUE ;
        }

        // Fill and send out a new buffer

        FillBuffer (((PWAVEHDR) lParam) -> lpData, iFreq) ;
        waveOutWrite (hWaveOut, (PWAVEHDR) lParam, sizeof (WAVEHDR)) ;
```

```
            return TRUE ;

        case MM_WOM_CLOSE:
            waveOutUnprepareHeader (hWaveOut, pWaveHdr1, sizeof (WAVEHDR)) ;
            waveOutUnprepareHeader (hWaveOut, pWaveHdr2, sizeof (WAVEHDR)) ;

            free (pWaveHdr1) ;
            free (pWaveHdr2) ;
            free (pBuffer1) ;
            free (pBuffer2) ;

            hWaveOut = NULL ;
            SetDlgItemText (hwnd, IDC_ONOFF, TEXT ("Turn On")) ;

            if (bClosing)
                EndDialog (hwnd, 0) ;

            return TRUE ;

        case WM_SYSCOMMAND:
            switch (wParam)
            {
            case SC_CLOSE:
                if (hWaveOut != NULL)
                {
                    bShutOff = TRUE ;
                    bClosing = TRUE ;

                    waveOutReset (hWaveOut) ;
                }
                else
                    EndDialog (hwnd, 0) ;

                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}
```

SINEWAVE.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog
```

```
SINEWAVE DIALOG DISCARDABLE 100, 100, 200, 50
STYLE WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Sine Wave Generator"
FONT 8, "MS Sans Serif"
BEGIN
    SCROLLBAR           IDC_SCROLL,8,8,150,12
    RTEXT               "440",IDC_TEXT,160,10,20,8
    LTEXT               "Hz",IDC_STATIC,182,10,12,8
    PUSHBUTTON          "Turn On",IDC_ONOFF,80,28,40,14
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by SineWave.rc

# define IDC_STATIC                 -1
# define IDC_SCROLL                 1000
# define IDC_TEXT                   1001
# define IDC_ONOFF                  1002
```

注意:FillBuffer 例程中用到的 OUT_BUFFER_SIZE、SAMPLE_RATE 和 PI 标识符在程序的顶部定义。FillBuffer 的 iFreq 参数是需要的频率,单位是 Hz。还要注意,sin 函数的结果调整到 0~254 的范围之间。对于每个样本,sin 函数的 fAngle 参数都增加一个值,该值的大小是 2π 弧度乘以需要的频率,再除以采样速率。

SINEWAVE 的窗口包含三个控件:一个用于选择频率的水平滚动条,一个用于显示当前所选频率的静态文本区域,以及一个标记为 Turn On 的按钮。按下此按钮后,将从连接声卡的扬声器中听到正弦波的声音,同时按钮上的文本将变成 Turn Off。用键盘或者鼠标移动滚动条可以改变频率。要关闭声音,可以再次按下该按钮。

SINEWAVE 代码初始化滚动条,以便频率在 WM_INITDIALOG 消息期间最低是 20 Hz,最高是 5000 Hz。初始化时滚动条设置为 440 Hz。用音乐术语来说就是中音上面的 A,它在管弦乐队演奏时用来调音。DlgProc 在接收 WM_HSCROLL 消息期间改变静态变量 iFreq。注意,Page Left 和 Page Right 将导致 DlgProc 增加或者减少一个八度音阶。

当 DlgProc 从按钮收到一个 WM_COMMAND 消息时,它首先分配 4 个内存块——两个用于 WAVEHDR 结构,我们马上讨论;另两个用于缓冲区保存波形数据,我们将这两个缓冲区称为 pBuffer1 和 pBuffer2。

通过调用 waveOutOpen 函数,SINEWAVE 打开波形音频设备以便输出,waveOutOpen 函数使用下面的参数:

```
waveOutOpen (&hWaveOut, wDeviceID, &waveformat, dwCallBack,
             dwCallBackData, dwFlags) ;
```

将第一个参数设置为指向 HWAVEOUT(handle to waveform audio output:波形音频输出句柄)类型的变量。从函数返回时,此变量将设置为一个句柄,后面的波形输出调用中将使用该句柄。

waveOutOpen 的第 2 个参数是设备 ID。它允许函数可以在安装多个声卡的机器上使用。参数的范围在 0 到系统所安装的波形输出设备数之间。调用 waveOutGetNumDevs 可以获得波形输出设备数,而调用 waveOutGetDevCaps 可以找出每个波形输出设备。如果想消除设备问号,那么您可以用常数 WAVE_MAPPER(定义为 -1)来选择设备,该设备在"控制面板"的"多媒体"中"音频"选项卡里的"首选设备"中指定。另外,如果首选设备不能满足您的需要,而其他设备可以,那么系统将选择其他设备。

第 3 个参数是指向 WAVEFORMATEX 结构的指针(后面将详细介绍)。第 4 个参数是窗口句柄或指向动态链接库中回调函数的指针,用来表示接收波形输出消息的窗口或者回调函数。使用回调函数时,可在第 5 个参数中指定程序定义的数据。dwFlags 参数可设为 CALLBACK_WINDOW 或 CALLBACK_FUNCTION,以表明第 4 个参数的类型。您也可用 WAVE_FORMAT_QUERY 标记来检查能否打开设备(实际上并不打开它)。还有其他几个标记可用。

waveOutOpen 的第 3 个参数定义为指向 WAVEFORMATEX 类型结构的指针,此结构在 MMSYSTEM.H 中定义如下:

```
typedef struct waveformat_tag
{
    WORD    wFormatTag ;          // waveform format = WAVE_FORMAT_PCM
    WORD    nChannels ;           // number of channels = 1 or 2
    DWORD   nSamplesPerSec ;      // sample rate
    DWORD   nAvgBytesPerSec ;     // bytes per second
    WORD    nBlockAlign ;         // block alignment
    WORD    wBitsPerSample ;      // bits per samples = 8 or 16
    WORD    cbSize ;              // 0 for PCM
}
WAVEFORMATEX, * PWAVEFORMATEX ;
```

您可用此结构指定采样速率(nSamplesPerSec)和采样精度(nBitsPerSample),以及选择单声道或立体声(nChannels)。结构中有些信息看起来是多余的,但该结构也可用于非 PCM 的采样方式。在非 PCM 采样方式下,此结构的最后一个字段设置为非 0 值,并带有其他信息。

对于 PCM 采样方式,nBlockAlign 字段设置为 nChannels 乘以 wBitsPerSample 再除以 8 所得到的数,它表示每次采样的总字节数;nAvgBytesPerSec 字段设置为 nSamplesPerSec 和 nBlockAlign 的乘积。

SINEWAVE 初始化 WAVEFORMATEX 结构的字段,并调用 waveOutOpen 函数:

```
waveOutOpen (&hWaveOut, WAVE_MAPPER, &waveformat,
             (DWORD) hwnd, 0, CALLBACK_WINDOW)
```

如果调用成功，则 waveOutOpen 函数返回 MMSYSERR_NOERROR(定义为 0)，否则返回非 0 的错误代码。如果 waveOutOpen 的返回值非 0，则 SINEWAVE 清除窗口，并显示一个标识错误的消息框。

现在设备打开了，SINEWAVE 通过初始化两个 WAVEHDR 结构的字段而继续，这两个结构用于穿过 API 传递缓冲区。WAVEHDR 定义如下：

```
typedef struct wavehdr_tag
{
    LPSTR lpData;                   // pointer to data buffer
    DWORD dwBufferLength;           // length of data buffer
    DWORD dwBytesRecorded;          // used for recorded
    DWORD dwUser;                   // for program use
    DWORD dwFlags;                  // flags
    DWORD dwLoops;                  // number of repetitions
    struct wavehdr_tag FAR * lpNext;  // reserved
    DWORD reserved;                 // reserved
}
WAVEHDR, * PWAVEHDR ;
```

SINEWAVE 将 lpData 字段设置为包含数据的缓冲区地址，dwBufferLength 字段设置为此缓冲区的大小，dwLoops 字段设置为 1，其他字段都设置为 0 或 NULL。如果要重复循环播放声音，可设置 dwFlags 和 dwLoops 字段。

接着 SINEWAVE 为两个信息头调用 waveOutPrepareHeader 函数，以防止结构和缓冲区与磁盘发生数据交换。

到此为止，所有的这些准备都是响应单击开启声音的按钮。但在程序的消息队列里已经有一个消息在等待响应。因为我们已经在函数 waveOutOpen 中指定要用一个窗口过程来接收波形输出消息，所以 waveOutOpen 函数向程序的消息队列发送了 MM_WOM_OPEN 消息，wParam 消息参数设置为波形输出句柄。要处理 MM_WOM_OPEN 消息，SINEWAVE 调用 FillBuffer 函数两次，并用正弦波形数据填充 pBuffer 缓冲区，然后 SINEWAVE 把两个 WAVEHDR 结构传送给 waveOutWrite，此函数将数据传送到波形输出硬件，才真正开始播放声音。

当波形硬件播放完 waveOutWrite 函数传送来的数据后，就向窗口发送 MM_WOM_DONE 消息，其中 wParam 参数是波形输出句柄，lParam 是指向 WAVEHDR 结构的指针。SINEWAVE 通过计算缓冲区的新数值，处理这一消息时，并通过调用 waveOutWrite 来重新提交缓冲区。

编写 SINEWAVE 程序时也可以只用一个 WAVEHDR 结构和一个缓冲区。不过，这样在播放完数据后将有一个很短暂的停顿，以等待程序处理 MM_WOM_DONE 消息来

提交新的缓冲区。SINEWAVE 使用的“双缓冲”技术避免了声音的不连续。

当用户单击 Turn Off 按钮关闭声音时，DlgProc 接收到另一个 WM_COMMAND 消息。对此消息，DlgProc 把 bShutOff 变量设置为 TRUE，并调用 waveOutReset 函数。此函数停止处理声音并发送一个 MM_WOM_DONE 消息。bShutOff 为 TRUE 时，SINEWAVE 通过调用 waveOutClose 来处理 MM_WOM_DONE，从而产生一个 MM_WOM_CLOSE 消息。处理 MM_WOM_CLOSE 通常包括清除过程。SINEWAVE 为两个 WAVEHDR 结构调用 waveOutUnprepareHeader，释放所有的内存块，并把按钮上的文字改回“Turn On”。

如果波形硬件继续播放缓冲区的声音数据，那么它自己调用 waveOutClose 将失效。您必须先调用 waveOutReset 来停止播放并产生 MM_WOM_DONE 消息。当 wParam 是 SC_CLOSE 时，DlgProc 也处理 WM_SYSCOMMAND 消息，这是因为用户从系统菜单中选择了 Close。如果波形音频继续播放，DlgProc 则调用 waveOutReset。无论如何，最后总要调用 EndDialog 来关闭对话框及结束程序。

22.2.6　数字录音机

Windows 提供了一个被称为“录音机”的程序来录制和播放数字声音。程序 22-3 (RECORD1)不如“录音机”完善，因为它没有包括任何文件 I/O，也不允许编辑声音。然而，这个程序给出了使用低级波形音频 API 来录制和回放声音的基本方法。

程序 22-3　RECORD1 程序

RECORD1.C

```
/*-------------------------
   RECORD1.C——Waveform Audio Recorder
              (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include "resource.h"

#define INP_BUFFER_SIZE 16384

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("Record1") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     if (-1 == DialogBox (hInstance, TEXT ("Record"), NULL, DlgProc))
```

```
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
    }
    return 0 ;
}

void ReverseMemory (BYTE * pBuffer, int iLength)
{
    BYTE b ;
    int  i ;

    for (i = 0 ; i < iLength / 2 ; i++)
    {
        b = pBuffer [i] ;
        pBuffer [i] = pBuffer [iLength - i - 1] ;
        pBuffer [iLength - i - 1] = b ;
    }
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static BOOL         bRecording, bPlaying, bReverse, bPaused,
                        bEnding, bTerminating ;
    static DWORD        dwDataLength, dwRepetitions = 1 ;
    static HWAVEIN      hWaveIn ;
    static HWAVEOUT     hWaveOut ;
    static PBYTE        pBuffer1, pBuffer2, pSaveBuffer, pNewBuffer ;
    static PWAVEHDR     pWaveHdr1, pWaveHdr2 ;
    static TCHAR        szOpenError[] = TEXT ("Error opening waveform audio!");
    static TCHAR        szMemError [] = TEXT ("Error allocating memory!") ;
    static WAVEFORMATEX waveform ;

    switch (message)
    {
    case WM_INITDIALOG:
            // Allocate memory for wave header

        pWaveHdr1 = malloc (sizeof (WAVEHDR)) ;
        pWaveHdr2 = malloc (sizeof (WAVEHDR)) ;

            // Allocate memory for save buffer

        pSaveBuffer = malloc (1) ;
        return TRUE ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
```

```
    }
    case IDC_RECORD_BEG:
            // Allocate buffer memory

        pBuffer1 = malloc (INP_BUFFER_SIZE) ;
        pBuffer2 = malloc (INP_BUFFER_SIZE) ;

        if (!pBuffer1 || !pBuffer2)
        {
            if (pBuffer1) free (pBuffer1) ;
            if (pBuffer2) free (pBuffer2) ;

            MessageBeep (MB_ICONEXCLAMATION) ;
            MessageBox (hwnd, szMemError, szAppName,
                        MB_ICONEXCLAMATION | MB_OK) ;
            return TRUE ;
        }

            // Open waveform audio for input

        waveform.wFormatTag      = WAVE_FORMAT_PCM ;
        waveform.nChannels       = 1 ;
        waveform.nSamplesPerSec  = 11025 ;
        waveform.nAvgBytesPerSec = 11025 ;
        waveform.nBlockAlign     = 1 ;
        waveform.wBitsPerSample  = 8 ;
        waveform.cbSize          = 0 ;

        if (waveInOpen (&hWaveIn, WAVE_MAPPER, &waveform,
                    (DWORD) hwnd, 0, CALLBACK_WINDOW))
        {
            free (pBuffer1) ;
            free (pBuffer2) ;
            MessageBeep (MB_ICONEXCLAMATION) ;
            MessageBox (hwnd, szOpenError, szAppName,
                        MB_ICONEXCLAMATION | MB_OK) ;
        }
            // Set up headers and prepare them

        pWaveHdr1 -> lpData          = pBuffer1 ;
        pWaveHdr1 -> dwBufferLength  = INP_BUFFER_SIZE ;
        pWaveHdr1 -> dwBytesRecorded = 0 ;
        pWaveHdr1 -> dwUser          = 0 ;
        pWaveHdr1 -> dwFlags         = 0 ;
        pWaveHdr1 -> dwLoops         = 1 ;
        pWaveHdr1 -> lpNext          = NULL ;
        pWaveHdr1 -> reserved        = 0 ;
```

```
        waveInPrepareHeader (hWaveIn, pWaveHdr1, sizeof (WAVEHDR)) ;

        pWaveHdr2 -> lpData              = pBuffer2 ;
        pWaveHdr2 -> dwBufferLength      = INP_BUFFER_SIZE ;
        pWaveHdr2 -> dwBytesRecorded     = 0 ;
        pWaveHdr2 -> dwUser              = 0 ;
        pWaveHdr2 -> dwFlags             = 0 ;
        pWaveHdr2 -> dwLoops             = 1 ;
        pWaveHdr2 -> lpNext              = NULL ;
        pWaveHdr2 -> reserved            = 0 ;

        waveInPrepareHeader (hWaveIn, pWaveHdr2, sizeof (WAVEHDR)) ;
        return TRUE ;

    case IDC_RECORD_END:
            // Reset input to return last buffer

        bEnding = TRUE ;
        waveInReset (hWaveIn) ;
        return TRUE ;

    case IDC_PLAY_BEG:
            // Open waveform audio for output

        waveform.wFormatTag              = WAVE_FORMAT_PCM ;
        waveform.nChannels               = 1 ;
        waveform.nSamplesPerSec          = 11025 ;
        waveform.nAvgBytesPerSec         = 11025 ;
        waveform.nBlockAlign             = 1 ;
        waveform.wBitsPerSample          = 8 ;
        waveform.cbSize                  = 0 ;

        if (waveOutOpen (&hWaveOut, WAVE_MAPPER, &waveform,
                         (DWORD) hwnd, 0, CALLBACK_WINDOW))
        {
            MessageBeep (MB_ICONEXCLAMATION) ;
            MessageBox (hwnd, szOpenError, szAppName,
                MB_ICONEXCLAMATION | MB_OK) ;
        }
        return TRUE ;

    case IDC_PLAY_PAUSE:
            // Pause or restart output

        if (!bPaused)
        {
            waveOutPause (hWaveOut) ;
```

```
                SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Resume")) ;
                bPaused = TRUE ;
            }
            else
            {
                waveOutRestart (hWaveOut) ;
                SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Pause")) ;
                bPaused = FALSE ;
            }
            return TRUE ;

        case IDC_PLAY_END:
                // Reset output for close preparation

            bEnding = TRUE ;
            waveOutReset (hWaveOut) ;
            return TRUE ;

        case IDC_PLAY_REV:
                // Reverse save buffer and play

            bReverse = TRUE ;
            ReverseMemory (pSaveBuffer, dwDataLength) ;

            SendMessage (hwnd, WM_COMMAND, IDC_PLAY_BEG, 0) ;
            return TRUE ;

        case IDC_PLAY_REP:
                // Set infinite repetitions and play

            dwRepetitions = -1 ;
            SendMessage (hwnd, WM_COMMAND, IDC_PLAY_BEG, 0) ;
            return TRUE ;

        case IDC_PLAY_SPEED:
                // Open waveform audio for fast output

            waveform.wFormatTag      = WAVE_FORMAT_PCM ;
            waveform.nChannels       = 1 ;
            waveform.nSamplesPerSec  = 22050 ;
            waveform.nAvgBytesPerSec = 22050 ;
            waveform.nBlockAlign     = 1 ;
            waveform.wBitsPerSample  = 8 ;
            waveform.cbSize          = 0 ;
            if (waveOutOpen (&hWaveOut, 0, &waveform, (DWORD) hwnd, 0,
                             CALLBACK_WINDOW))
            {
```

```
                MessageBeep (MB_ICONEXCLAMATION) ;
                MessageBox (hwnd, szOpenError, szAppName,
                                MB_ICONEXCLAMATION | MB_OK) ;
            }
            return TRUE ;
        }
        break ;

    case MM_WIM_OPEN:
            // Shrink down the save buffer

        pSaveBuffer = realloc (pSaveBuffer, 1) ;

            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG),    FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END),    TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),      FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),    FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),      FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REV),      FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REP),      FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SPEED),    FALSE) ;
        SetFocus (GetDlgItem (hwnd, IDC_RECORD_END)) ;

            // Add the buffers

        waveInAddBuffer (hWaveIn, pWaveHdr1, sizeof (WAVEHDR)) ;
        waveInAddBuffer (hWaveIn, pWaveHdr2, sizeof (WAVEHDR)) ;

            // Begin sampling

        bRecording = TRUE ;
        bEnding = FALSE ;
        dwDataLength = 0 ;
        waveInStart (hWaveIn) ;
        return TRUE ;

    case MM_WIM_DATA:

            // Reallocate save buffer memory

        pNewBuffer = realloc (pSaveBuffer, dwDataLength +
                              ((PWAVEHDR) lParam) -> dwBytesRecorded) ;

        if (pNewBuffer == NULL)
        {
```

```
            waveInClose (hWaveIn) ;
            MessageBeep (MB_ICONEXCLAMATION) ;
            MessageBox (hwnd, szMemError, szAppName,
                        MB_ICONEXCLAMATION | MB_OK) ;
            return TRUE ;
        }

        pSaveBuffer = pNewBuffer ;
        CopyMemory (pSaveBuffer + dwDataLength, ((PWAVEHDR) lParam) -> lpData,
                    ((PWAVEHDR) lParam) -> dwBytesRecorded) ;

        dwDataLength += ((PWAVEHDR) lParam) -> dwBytesRecorded ;

        if (bEnding)
        {
            waveInClose (hWaveIn) ;
            return TRUE ;
        }

            // Send out a new buffer

        waveInAddBuffer (hWaveIn, (PWAVEHDR) lParam, sizeof (WAVEHDR)) ;
        return TRUE ;

    case MM_WIM_CLOSE:
            // Free the buffer memory

        waveInUnprepareHeader (hWaveIn, pWaveHdr1, sizeof (WAVEHDR)) ;
        waveInUnprepareHeader (hWaveIn, pWaveHdr2, sizeof (WAVEHDR)) ;

        free (pBuffer1) ;
        free (pBuffer2) ;

            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), FALSE) ;
        SetFocus (GetDlgItem (hwnd, IDC_RECORD_BEG)) ;

        if (dwDataLength > 0)
        {
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),    TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),  FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),    FALSE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REP),    TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REV),    TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SPEED),  TRUE) ;
```

```
            SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;
        }
        bRecording = FALSE ;

        if (bTerminating)
            SendMessage (hwnd, WM_SYSCOMMAND, SC_CLOSE, 0L) ;

        return TRUE ;

    case MM_WOM_OPEN:
            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG),   FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END),   FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),     FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),   TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),     TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REP),     FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REV),     FALSE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SPEED),   FALSE) ;
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_END)) ;

            // Set up header

        pWaveHdr1 -> lpData             = pSaveBuffer ;
        pWaveHdr1 -> dwBufferLength     = dwDataLength ;
        pWaveHdr1 -> dwBytesRecorded    = 0 ;
        pWaveHdr1 -> dwUser             = 0 ;
        pWaveHdr1 -> dwFlags            = WHDR_BEGINLOOP | WHDR_ENDLOOP ;
        pWaveHdr1 -> dwLoops            = dwRepetitions ;
        pWaveHdr1 -> lpNext             = NULL ;
        pWaveHdr1 -> reserved           = 0 ;

            // Prepare and write

        waveOutPrepareHeader (hWaveOut, pWaveHdr1, sizeof (WAVEHDR)) ;
        waveOutWrite (hWaveOut, pWaveHdr1, sizeof (WAVEHDR)) ;

        bEnding = FALSE ;
        bPlaying = TRUE ;
        return TRUE ;

    case MM_WOM_DONE:
        waveOutUnprepareHeader (hWaveOut, pWaveHdr1, sizeof (WAVEHDR)) ;
        waveOutClose (hWaveOut) ;
        return TRUE ;
```

```
case MM_WOM_CLOSE:
        // Enable and disable buttons

     EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG),   TRUE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END),   TRUE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),     TRUE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),   FALSE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),     FALSE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REV),     TRUE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_REP),     TRUE) ;
     EnableWindow (GetDlgItem (hwnd, IDC_PLAY_SPEED),   TRUE) ;
     SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

     SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Pause")) ;
     bPaused = FALSE ;
     dwRepetitions = 1 ;
     bPlaying = FALSE ;

     if (bReverse)
     {
          ReverseMemory (pSaveBuffer, dwDataLength) ;
          bReverse = FALSE ;
     }

     if (bTerminating)
          SendMessage (hwnd, WM_SYSCOMMAND, SC_CLOSE, 0L) ;

     return TRUE ;

case WM_SYSCOMMAND:
     switch (LOWORD (wParam))
     {
     case SC_CLOSE:
          if (bRecording)
          {
               bTerminating = TRUE ;
               bEnding = TRUE ;
               waveInReset (hWaveIn) ;
               return TRUE ;
          }
          if (bPlaying)
          {
               bTerminating = TRUE ;
               bEnding = TRUE ;
               waveOutReset (hWaveOut) ;
               return TRUE ;
          }
```

```
                free (pWaveHdr1) ;
                free (pWaveHdr2) ;
                free (pSaveBuffer) ;
                EndDialog (hwnd, 0) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}
```

RECORD.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog

RECORD DIALOG DISCARDABLE 100, 100, 152, 74
STYLE WS_MINIMIZEBOX | WS_VISIBLE | WS_CAPTION | WS_SYSMENU
CAPTION "Waveform Audio Recorder"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Record",IDC_RECORD_BEG,28,8,40,14
    PUSHBUTTON      "End",IDC_RECORD_END,76,8,40,14,WS_DISABLED
    PUSHBUTTON      "Play",IDC_PLAY_BEG,8,30,40,14,WS_DISABLED
    PUSHBUTTON      "Pause",IDC_PLAY_PAUSE,56,30,40,14,WS_DISABLED
    PUSHBUTTON      "End",IDC_PLAY_END,104,30,40,14,WS_DISABLED
    PUSHBUTTON      "Reverse",IDC_PLAY_REV,8,52,40,14,WS_DISABLED
    PUSHBUTTON      "Repeat",IDC_PLAY_REP,56,52,40,14,WS_DISABLED
    PUSHBUTTON      "Speedup",IDC_PLAY_SPEED,104,52,40,14,WS_DISABLED
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by Record.rc

# define IDC_RECORD_BEG                 1000
# define IDC_RECORD_END                 1001
# define IDC_PLAY_BEG                   1002
# define IDC_PLAY_PAUSE                 1003
# define IDC_PLAY_END                   1004
# define IDC_PLAY_REV                   1005
```

```
# define IDC_PLAY_REP          1006
# define IDC_PLAY_SPEED        1007
```

RECORD.RC 和 RESOURCE.H 文件也在 RECORD2 和 RECORD3 程序中使用。

RECORD1 窗口共有 8 个按钮。第一次运行 RECORD1 时，只有 Record 按钮有效。按下 Record 后，就开始录音，这时 Record 按钮无效，而 End 按钮有效。按下 End 可停止录音。这时，Play、Reverse、Repeat 和 Speedup 按钮都有效，选择任一个按钮都可重放声音：Play 表示正常播放；Reverse 表示反向播放；Repeat 表示无限的重复播放（好像循环录音带）；Speedup 以正常速度的两倍来播放。您可以按下第二个 End 按钮来结束播放。而按下 Pause 按钮可暂停播放。按下后，Pause 按钮将变为 Resume 按钮，用于恢复播放声音。如果要录制另一段声音，则新录制的声音将替换内存里现有的声音。

任何时候，有效按钮都是可以执行有效操作的按钮。这需要在 RECORD1 源代码中包括对 EnableWindow 的多次调用，但是程序并不检查具体的按钮操作是否有效。显然，这使得程序操作更为直观。

RECORD1 用了许多快捷方式来简化代码。首先，如果安装了多个波形音频硬件设备，则 RECORD1 只使用默认设备。其次，程序按标准的 11.025 kHz 的采样速率和 8 位的采样精度来录音和放音，而不管设备能否提供更高的采样速率和采样精度。唯一的例外是加速功能，加速时 RECORD1 按 22.050 kHz 的采样速率播放声音，这样不仅播放速度提高了一倍，而且频率也提高了一个音阶。

录制声音既包括为输入而打开波形音频硬件，还包括将缓冲区传递给 API，以便接收声音数据。

RECORD1 设有几个内存块，其中三个很小，至少在初始化时很小，并且在 DlgProc 的 WM_INITDIALOG 消息期间进行分配。程序分配两个 WAVEHDR 结构，分别由指针 pWaveHdr1 和 pWaveHdr2 指向。这两个结构用于将缓冲区传递给波形 API。pSaveBuffer 指针指向存储整个录音的缓冲区，最初分配时只有一个字节。以后，随着录音的进行，该缓冲区不断增大，以适应所有的声音数据。（如果录音时间过长，则 RECORD1 能够在录制过程中及时发现内存溢出，并允许您重放成功存储的声音。）由于这个缓冲区用来存储堆积的声音数据，所以我将其称为“存储缓冲区（save buffer）”。指针 pBuffer1 和 pBuffer2 指向的另外两个内存块，大小是 16 K，它们在记录接收的声音数据时分配，录音结束后释放这些内存块。

8 个按钮中的每一个都向 REPORT1 窗口的对话过程 DlgProc 产生 WM_COMMAND 消息。最初只有 Record 按钮有效。按下此按钮将产生 WM_COMMAND 消息，其中 wParam 参数等于 IDC_RECORD_BEG。为处理这个消息，RECORD1 分配两个 16 K 的缓冲区来接收声音数据，初始化 WAVEFORMATEX 结构的字段，并将此结构传递给 waveInOpen 函数，然后设置两个 WAVEHDR 结构。

waveInOpen 函数产生一个 MM_WIM_OPEN 消息。在此消息期间，RECORD1 把存储缓冲区的大小缩减到 1 个字节，以准备接收数据。（当然，第一次录音时，存储缓冲区的大小就是 1 个字节，但以后录制时，就可能大多了。）在 MM_WIM_OPEN 消息期间，RECORD1 也将适当的按钮设置为有效和无效，然后，程序用 waveInAddBuffer 把两个 WAVEHDR 结构和缓冲区传送给 API。这时会设置某些标记，然后调用 waveInStart 开始录音。

采用 11.025 kHz 的采样速率和 8 位的采样精度时，16 K 的缓冲区可存储大约1.5秒的声音。这时，RECORD1 接收 MM_WIM_DATA 消息。在响应此消息期间，程序将根据变量 dwDataLength 和 WAVEHDR 结构中的字段 dwBytesRecorded 对缓冲区重新分配。如果分配失败，RECORD1 调用 waveInClose 来停止录音。

如果重新分配成功，则 RECORD1 把 16 K 缓冲区里的数据复制到存储缓冲区，然后再次调用 waveInAddBuffer。此过程将持续到 RECORD1 用完保存缓冲区的内存，或用户按下 End 按钮为止。

End 按钮产生 WM_COMMAND 消息，其中 wParam 等于 IDC_RECORD_END。处理这个消息很简单，RECORD1 把 bEnding 标记设置为 TRUE，并调用 waveInReset。waveInReset 函数使录音停止，并产生 MM_WIM_DATA 消息，该消息含有部分填充的缓冲区。除了调用 waveInClose 来关闭波形输入设备外，RECORD1 对该消息正常响应。

waveInClose 产生 MM_WIM_CLOSE 消息。RECORD1 响应此消息时，释放 16 K 输入缓冲区，并使相应的按钮有效或无效。尤其是，当存储缓冲区里存有数据（除非第一次分配就失败，否则一般都含有数据）时，播放按钮将有效。

录音以后，存储缓冲区里将含有这些声音数据。当用户选择 Play 按钮时，DlgProc 就接收一个 WM_COMMAND 消息，其中 wParam 等于 IDC_PLAY_BEG。响应时，程序将初始化 WAVEFORMATEX 结构的字段，并调用 waveOutOpen。

waveOutOpen 调用再次产生 MM_WOM_OPEN 消息，在此消息期间，RECORD1 把相应的按钮置为有效或无效（只允许使用 Pause 和 End），用存储缓冲区来初始化 WAVEHDR 结构的字段，调用 waveOutPrepareHeader 来准备要播放的声音，然后调用 waveOutWrite 开始播放。

一般情况下，直到播放完存储缓冲区里的所有数据才停止。这时产生 MM_WOM_DONE 消息。如果还有缓冲区要播放，则程序会在这时将它们传递给 API。由于 RECORD1 只播放一个大缓冲区，因此程序不再简单地准备标题，而是调用 waveOutClose。waveOutClose 函数产生 MM_WOM_CLOSE 消息。在此消息期间，RECORD1 使相应的按钮有效或无效，并允许声音再次播放或者录制新声音。

程序中还有一个 End 按钮，利用此按钮，用户可以在播放完存储缓冲区之前的任意时刻停止播放。End 按钮产生一个 WM_COMMAND 消息，其中 wParam 等于 IDC_PLAY_

END，响应时，程序调用 waveOutReset，此函数产生一个正常处理的 MM_WOM_DONE 消息。

RECORD1 的窗口中还包括一个 Pause 按钮。处理此按钮很简单：第一次按下，RECORD1 调用 waveOutPause 来暂停播放，并将 Pause 按钮上的文字改为 Resume。按下 Resume 按钮时，通过调用 waveOutRestart 来继续播放。

为了使程序更有趣，窗口中还包括另外三个按钮：Reverse、Repeat 和 Speedup。这些按钮都产生 WM_COMMAND 消息，其中 wParam 的值分别等于 IDC_PLAY_REV、IDC_PLAY_REP 和 IDC_PLAY_SPEED。

反向播放声音就是把存储缓冲区里的数据按字节顺序反向，然后再正常播放。RECORD1 中有一个称为 ReverseMemory 的小函数使字节反向。在 WM_COMMAND 消息期间，程序在播放块之前调用此函数，并在 MM_WOM_CLOSE 消息的后期再次调用此函数，以便将其恢复到正常状态。

Repeat 按钮将往复不停地播放声音。由于 API 支持重复播放声音，所以这并不复杂。只要将 WAVEHDR 结构的 dwLoops 字段设为重复次数，将 dwFlags 字段设为 WHDR_BEGINLOOP 和 WHDR_ENDLOOP，分别表示循环时缓冲区的开始部分和结束部分。因为 RECORD1 只使用一个缓冲区来播放声音，所以这两个标记被连结到 dwFlags 字段中。

要实现两倍速播放也很容易。在准备为输出而打开波形音频期间，初始化 WAVEFORMATEX 结构的字段时，只需将 nSamplesPerSec 和 nAvgBytesPerSec 字段设置为 22 050，而不是 11 025。

22.2.7　另一种 MCI

您可能已经发现，RECORD1 很复杂，特别是在处理波形音频函数调用及其产生的消息间的交互时更复杂。处理可能出现的内存不足的情况也是如此，但这也许正是它称为低级接口的原因。在本章的前面提到过，Windows 也提供高级媒体控制接口（Media Control Interface）。

对波形音频来说，低级接口与 MCI 之间的主要区别在于 MCI 用波形文件记录声音数据，并通过读取文件来播放声音。由于在播放声音之前要读取文件、处理文件，然后再写入文件，所以让 RECODE1 来实现“特殊效果”很困难。这是典型的折中方法：功能齐全还是使用方便。低级接口很灵活，但 MCI（其中的大部分）更方便。

MCI 有两种不同但又相关的实现形式。一种形式用消息和数据结构将命令发送给多媒体设备，然后再从那里接收信息；另一种形式使用 ASCII 文本字符串。创建基于文本的接口最初是为了允许多媒体设备接受简单的脚本语言的控制，但它也提供非常容易的交互控制，请参见本章前面 TESTMCI 程序的演示。

RECORD2 程序，如程序 22-4 所示，使用 MCI 形式的消息和数据结构来实现另一个

数字音频录音机和播放器。虽然它使用的对话框模板与 RECORD1 一样，但并没有实现三个特殊效果的按钮。

程序 22-4　RECORD2 程序

RECORD2.C

```
/*--------------------------
   RECORD2.C——Waveform Audio Recorder
              (c) Charles Petzold, 1998
   --------------------------*/

#include <windows.h>
#include "..\\record1\\resource.h"

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("Record2") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     if (-1 == DialogBox (hInstance, TEXT ("Record"), NULL, DlgProc))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
     }
     return 0 ;
}

void ShowError (HWND hwnd, DWORD dwError)
{
     TCHAR szErrorStr [1024] ;

     mciGetErrorString (dwError, szErrorStr,
                        sizeof (szErrorStr) / sizeof (TCHAR)) ;
     MessageBeep (MB_ICONEXCLAMATION) ;
     MessageBox (hwnd, szErrorStr, szAppName, MB_OK | MB_ICONEXCLAMATION) ;
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BOOL             bRecording, bPlaying, bPaused ;
     static TCHAR            szFileName[] = TEXT ("record2.wav") ;
     static WORD             wDeviceID ;
     DWORD                   dwError ;
     MCI_GENERIC_PARMS       mciGeneric ;
```

```
MCI_OPEN_PARMS      mciOpen ;
MCI_PLAY_PARMS      mciPlay ;
MCI_RECORD_PARMS    mciRecord ;
MCI_SAVE_PARMS      mciSave ;

switch (message)
{
case WM_COMMAND:
    switch (wParam)
    {
    case IDC_RECORD_BEG:
            // Delete existing waveform file

        DeleteFile (szFileName) ;

            // Open waveform audio

        mciOpen.dwCallback          = 0 ;
        mciOpen.wDeviceID           = 0 ;
        mciOpen.lpstrDeviceType     = TEXT ("waveaudio") ;
        mciOpen.lpstrElementName    = TEXT ("") ;
        mciOpen.lpstrAlias          = NULL ;
        dwError = mciSendCommand (0, MCI_OPEN,
                          MCI_WAIT | MCI_OPEN_TYPE | MCI_OPEN_ELEMENT,
                          (DWORD) (LPMCI_OPEN_PARMS) &mciOpen) ;
        if (dwError != 0)
        {
             ShowError (hwnd, dwError) ;
             return TRUE ;
        }
            // Save the Device ID

        wDeviceID = mciOpen.wDeviceID ;

            // Begin recording

        mciRecord.dwCallback = (DWORD) hwnd ;
        mciRecord.dwFrom     = 0 ;
        mciRecord.dwTo       = 0 ;

        mciSendCommand (wDeviceID, MCI_RECORD, MCI_NOTIFY,
                        (DWORD) (LPMCI_RECORD_PARMS) &mciRecord) ;

            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), TRUE) ;
```

```
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),    FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),  FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),    FALSE);
        SetFocus (GetDlgItem (hwnd, IDC_RECORD_END)) ;

        bRecording = TRUE ;
        return TRUE ;

    case IDC_RECORD_END:
            // Stop recording

        mciGeneric.dwCallback = 0 ;

        mciSendCommand (wDeviceID, MCI_STOP, MCI_WAIT,
                        (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

            // Save the file

        mciSave.dwCallback = 0 ;
        mciSave.lpfilename = szFileName ;

        mciSendCommand (wDeviceID, MCI_SAVE, MCI_WAIT | MCI_SAVE_FILE,
                        (DWORD) (LPMCI_SAVE_PARMS) &mciSave) ;

            // Close the waveform device

        mciSendCommand (wDeviceID, MCI_CLOSE, MCI_WAIT,
                        (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG),   TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END),   FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),     TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),   FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),     FALSE);
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

        bRecording = FALSE ;
        return TRUE ;

    case IDC_PLAY_BEG:
            // Open waveform audio

        mciOpen.dwCallback      = 0 ;
        mciOpen.wDeviceID       = 0 ;
```

```
        mciOpen.lpstrDeviceType       = NULL ;
        mciOpen.lpstrElementName      = szFileName ;
        mciOpen.lpstrAlias            = NULL ;

        dwError = mciSendCommand (0, MCI_OPEN,
                                  MCI_WAIT | MCI_OPEN_ELEMENT,
                                  (DWORD) (LPMCI_OPEN_PARMS) &mciOpen) ;

        if (dwError != 0)
        {
             ShowError (hwnd, dwError) ;
             return TRUE ;
        }
            // Save the Device ID

        wDeviceID = mciOpen.wDeviceID ;

            // Begin playing

        mciPlay.dwCallback = (DWORD) hwnd ;
        mciPlay.dwFrom     = 0 ;
        mciPlay.dwTo       = 0 ;

        mciSendCommand (wDeviceID, MCI_PLAY, MCI_NOTIFY,
                        (DWORD) (LPMCI_PLAY_PARMS) &mciPlay) ;

            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG),  FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END),  FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),    FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE),  TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),    TRUE) ;
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_END)) ;

        bPlaying = TRUE ;
        return TRUE ;

    case IDC_PLAY_PAUSE:
        if (!bPaused)
                // Pause the play
        {
             mciGeneric.dwCallback = 0 ;

             mciSendCommand (wDeviceID, MCI_PAUSE, MCI_WAIT,
                             (DWORD) (LPMCI_GENERIC_PARMS) & mciGeneric);
```

```
                SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Resume")) ;
                bPaused = TRUE ;
            }
            else
                // Begin playing again
            {
                mciPlay.dwCallback = (DWORD) hwnd ;
                mciPlay.dwFrom     = 0 ;
                mciPlay.dwTo       = 0 ;

                mciSendCommand (wDeviceID, MCI_PLAY, MCI_NOTIFY,
                                (DWORD) (LPMCI_PLAY_PARMS) &mciPlay) ;

                SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Pause")) ;
                bPaused = FALSE ;
            }

            return TRUE ;

        case IDC_PLAY_END:
                // Stop and close

            mciGeneric.dwCallback = 0 ;

            mciSendCommand (wDeviceID, MCI_STOP, MCI_WAIT,
                            (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

            mciSendCommand (wDeviceID, MCI_CLOSE, MCI_WAIT,
                            (DWORD) (LPMCI_GENERIC_PARMS) &mciGeneric) ;

                // Enable and disable buttons

            EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), FALSE);
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),   TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE), FALSE);
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),   FALSE);
            SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

            bPlaying = FALSE ;
            bPaused = FALSE ;
            return TRUE ;
        }
        break ;

    case MM_MCINOTIFY:
```

```
            switch (wParam)
            {
            case MCI_NOTIFY_SUCCESSFUL:
                if (bPlaying)
                    SendMessage (hwnd, WM_COMMAND, IDC_PLAY_END, 0) ;

                if (bRecording)
                    SendMessage (hwnd, WM_COMMAND, IDC_RECORD_END, 0);

                return TRUE ;
            }
            break ;

        case WM_SYSCOMMAND:
            switch (wParam)
            {
            case SC_CLOSE:
                if (bRecording)
                    SendMessage (hwnd, WM_COMMAND, IDC_RECORD_END, 0L) ;
                if (bPlaying)
                    SendMessage (hwnd, WM_COMMAND, IDC_PLAY_END, 0L) ;

                EndDialog (hwnd, 0) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}
```

RECORD2 只使用两个 MCI 函数调用，其中最重要的调用如下所示：

```
error = mciSendCommand (wDeviceID, message, dwFlags, dwParam)
```

第一个参数是设备的数字标识号，您可以按句柄来使用 ID 号。打开设备时就可以获得 ID，并在随后的 mciSendCommand 调用中使用。第二个参数是前缀为 MCI 的常数。这些称为“MCI 命令消息”，RECORD2 演示了其中的 7 个：MCI_OPEN、MCI_RECORD、MCI_STOP、MCI_SAVE、MCI_PLAY、MCI_PAUSE 和 MCI_CLOSE。

dwFlags 参数通常由 0 或者多个位标记常数（由 C 的位“或”操作符组成）组成，通常用来表示不同的选项。一些选项是某个命令消息所特有的，而另一些对所有的消息都是通用的。dwParam 参数通常是指向一个数据结构的长指针，该结构表明选项以及从设备获得的信息。许多 MCI 消息都与数据结构有关，而且这些数据结构对于消息来说都是唯一的。

如果 mciSendCommand 函数调用成功，则返回 0 值，否则返回报错代码。要向用户

报告此错误，可用下面的函数获得描述错误的文本字符串：

```
mciGetErrorString (error, szBuffer, dwLength)
```

此函数在程序 TESTMCI 中也用到过。

按下 Record 按钮后，RECORD2 的窗口过程就收到一个 WM_COMMAND 消息，其中 wParam 等于 IDC_RECORD_BEG。RECORD2 从打开设备开始，包括设置 MCI_OPEN_PARMS 结构的字段，并用 MCI_OPEN 命令消息调用 mciSendCommand。录音时，lpstrDeviceType 字段设置为字符串 waveaudio 以说明设备类型，lpstrElementName 字段设置为长度为 0 的字符串。MCI 驱动程序使用默认的采样速率和采样精度，但是您可以用 MCI_SET 命令进行修改。录音过程中，声音数据先保存在硬盘上的临时文件中，最后再转化成标准的波形文件。本章的后面将介绍波形文件的格式。播放录制的声音时，MCI 使用波形文件中定义的采样速率和采样精度。

如果 RECORD2 不能打开设备，则用 mciGetErrorString 和 MessageBox 提示错误信息；否则从 mciSendCommand 调用返回，MCI_OPEN_PARMS 结构的 wDeviceID 字段包含有设备 ID，以供后面的调用使用。

要开始录音，RECORD2 就调用 mciSendCommand，以 MCI_RECORD 命令消息和 MCI_WAVE_RECORD_PARMS 数据结构为参数。当然，您也可以将此结构（并使用表示这些字段已设置的位标记）的 dwFromz 和 dwTo 字段进行设置，以便将声音插入现有的波形文件，其文件名在 MCI_OPEN_PARMS 结构的 lpstrElementName 字段指定。默认情况下，任何新的声音都插入在现有文件的开始位置。

RECORD2 将 MCI_WAVE_RECORD_PARMS 结构的 dwCallback 字段设置为程序的窗口句柄，并在 mciSendCommand 调用中包含 MCI_NOTIFY 标记。这导致录音结束后向窗口过程发送一个通知消息。我将简要讨论一下这个通知消息。

录音结束后，按下前一个 End 按钮来停止录音，这时产生一个 WM_COMMAND 消息，其中 wParam 等于 IDC_RECORD_END。作为响应，窗口过程将调用 mciSendCommand 三次：MCI_STOP 命令消息用于停止录音；MCI_SAVE 命令消息用于把临时文件中的声音数据传递到 MCI_SAVE_PARMS 结构中指定的文件（“record2.wav”）；MCI_CLOSE 命令消息用于删除所有的临时文件、释放已经创建的内存块，并关闭设备。

播放时，MCI_OPEN_PARMS 结构的 lpstrElementName 字段设置为文件名“record2.wav”。mciSendCommand 第三个参数中所包含的 MCI_OPEN_ELEMENT 标记表示 lpstrElementName 字段是一个有效的文件名。通过文件的扩展名.WAV，MCI 知道用户要打开一个波形音频设备。如果存在多个波形硬件，则打开第一个。（设置 MCI_OPEN_PARMS 结构的 lpstrDeviceType 字段也可以打开其他波形设备。）

播放将包括带有 MCI_PLAY 命令消息和 MCI_PLAY_PARMS 结构的 mciSendCommand 调用。虽然波形文件的任意部分都可以播放，但 RECORD2 只播放整个文件。

RECORD2 还包括一个 Pause 按钮来暂停播放声音文件。这个按钮产生一个 WM_COMMAND 消息，其中 wParam 等于 IDC_PLAY_PAUSE。作为响应，程序将调用 mciSendCommand，并以 MCI_PAUSE 命令消息和 MCI_GENERIC_PARMS 结构作为参数。MCI_GENERIC_PARMS 结构用于这样一些消息：它们除了需要用于通知的可选窗口句柄外，不需要任何信息。如果播放已经暂停，则通过再次使用 MCI_PLAY 命令消息调用 mciSendCommand 继续播放。

按下第二个 End 按钮也可以停止播放。这时产生 wParam 等于 IDC_PLAY_END 的 WM_COMMAND 消息。作为响应，窗口过程将调用 mciSendCommand 两次：第一次使用 MCI_STOP 命令消息；第二次使用 MCI_CLOSE 命令消息。

现在有一个问题：虽然可以通过按下 End 按钮来手工终止播放，但您可能需要播放整个文件。程序如何知道文件播放完的时间呢？这是 MCI 通知消息的任务。

当带有 MCI_RECORD 和 MCI_PLAY 消息来调用 mciSendCommand 时，RECORD2 将包括 MCI_NOTIFY 标记，并将数据结构的 dwCallback 字段设置为程序窗口句柄。这样就产生一个通知消息，称为 MM_MCINOTIFY，并在某些环境下传递给窗口过程。消息参数 wParam 是一个状态代码，而 lParam 是设备 ID。

带有 MCI_STOP 或者 MCI_PAUSE 命令消息来调用 mciSendCommand 时，将接收到一个伴有 wParam 等于 MCI_NOTIFY_ABORTED 的 MM_MCINOTIFY 消息。当按下 Pause 按钮或者两个 End 按钮中的一个时，就会出现这种情况。由于对这些按钮已进行过适当的处理，所以 RECORD2 可以忽略这种情况。播放时，会在声音文件结束后接收到伴有 wParam 等于 MCI_NOTIFY_SUCCESSFUL 的 MM_MCINOTIFY 消息。这种情况下，窗口过程给自己发送一个 wParam 等于 IDC_PLAY_ENDWM_COMMAND 的消息来模拟用户按下 End 按钮，然后窗口过程做出正常响应：停止播放，关闭设备。

录音时，如果用于存储临时文件的硬盘空间不够，就会接收一个 MM_MCINOTIFY 消息，其中 wParam 等于 MCI_NOTIFY_SUCCESSFUL。（虽然现在还不能说它很完美，但其功能已经很齐全了。）作为响应，窗口过程给自己发送一个 WM_COMMAND 消息，其中 wParam 等于 IDC_RECORD_END，然后与正常情况下一样：停止录音、保存文件，然后关闭设备。

22.2.8 MCI 命令字符串的方法

Windows 的多媒体接口曾经包含函数 mciExecute，其语法如下：

```
bSuccess = mciExecute (szCommand) ;
```

其中唯一的参数是 MCI 命令字符串。函数返回布尔值——如果调用成功，则返回非 0 值，否则返回 0。在功能上，mciExecute 函数等价于调用后三个参数为 NULL 或 0 的 mciSendString(TESTMCI 中使用的基于字符串的 MCI 函数)，然后在发生错误时调用 mciGetErrorString 和 MessageBox。

虽然 mciExecute 不再是 API 的一部分，但我还是在 RECORD3 版的数字录音播放机中使用了这个函数。和 RECORD2 一样，RECORD3 程序也使用 RECORD1 中的资源描述文件 RECORD.RC 和 RESOURCE.H，如程序 22-5 所示。

程序 22-5　RECORD3 程序

RECORD3.C

```
/*-------------------------
   RECORD3.C——Waveform Audio Recorder
               (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include "..\\record1\\resource.h"

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("Record3") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     if (-1 == DialogBox (hInstance, TEXT ("Record"), NULL, DlgProc))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
     }
     return 0 ;
}

BOOL mciExecute (LPCTSTR szCommand)
{
     MCIERROR  error ;
     TCHAR     szErrorStr [1024] ;

     if (error = mciSendString (szCommand, NULL, 0, NULL))
     {
          mciGetErrorString (error, szErrorStr,
                             sizeof (szErrorStr) / sizeof (TCHAR)) ;
```

```
            MessageBeep (MB_ICONEXCLAMATION) ;
            MessageBox (NULL, szErrorStr, TEXT ("MCI Error"),
                        MB_OK | MB_ICONEXCLAMATION) ;
        }
        return error == 0 ;
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
        static BOOL bRecording, bPlaying, bPaused ;

        switch (message)
        {
        case WM_COMMAND:
            switch (wParam)
            {
            case IDC_RECORD_BEG:
                    // Delete existing waveform file

                DeleteFile (TEXT ("record3.wav")) ;

                    // Open waveform audio and record

                if (! mciExecute (TEXT ("open new type waveaudio alias mysound")))
                    return TRUE ;

                mciExecute (TEXT ("record mysound")) ;

                    // Enable and disable buttons

                EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), FALSE);
                EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), TRUE) ;
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),   FALSE);
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE), FALSE);
                EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),   FALSE);
                SetFocus (GetDlgItem (hwnd, IDC_RECORD_END)) ;

                bRecording = TRUE ;
                return TRUE ;

            case IDC_RECORD_END:
                    // Stop, save, and close recording

                mciExecute (TEXT ("stop mysound")) ;
                mciExecute (TEXT ("save mysound record3.wav")) ;
                mciExecute (TEXT ("close mysound")) ;
```

```
            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),   TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE), FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),   FALSE);
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

        bRecording = FALSE ;
        return TRUE ;

    case IDC_PLAY_BEG:
            // Open waveform audio and play

        if (! mciExecute (TEXT ("open record3.wav alias mysound")))
            return TRUE ;

        mciExecute (TEXT ("play mysound")) ;

            // Enable and disable buttons

        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),   FALSE);
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE), TRUE) ;
        EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),   TRUE) ;
        SetFocus (GetDlgItem (hwnd, IDC_PLAY_END)) ;

        bPlaying = TRUE ;
        return TRUE ;

    case IDC_PLAY_PAUSE:
        if (!bPaused)
                // Pause the play
        {
            mciExecute (TEXT ("pause mysound")) ;
            SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Resume")) ;
            bPaused = TRUE ;
        }
        else
                // Begin playing again
        {
            mciExecute (TEXT ("play mysound")) ;
            SetDlgItemText (hwnd, IDC_PLAY_PAUSE, TEXT ("Pause")) ;
            bPaused = FALSE ;
```

```
            }

            return TRUE ;

        case IDC_PLAY_END:
                // Stop and close

            mciExecute (TEXT ("stop mysound")) ;
            mciExecute (TEXT ("close mysound")) ;

                // Enable and disable buttons

            EnableWindow (GetDlgItem (hwnd, IDC_RECORD_BEG), TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_RECORD_END), FALSE);
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_BEG),   TRUE) ;
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_PAUSE), FALSE);
            EnableWindow (GetDlgItem (hwnd, IDC_PLAY_END),   FALSE);
            SetFocus (GetDlgItem (hwnd, IDC_PLAY_BEG)) ;

            bPlaying = FALSE ;
            bPaused = FALSE ;
            return TRUE ;
        }
        break ;

    case WM_SYSCOMMAND:
        switch (wParam)
        {
        case SC_CLOSE:
            if (bRecording)
                SendMessage (hwnd, WM_COMMAND, IDC_RECORD_END, 0L);

            if (bPlaying)
                SendMessage (hwnd, WM_COMMAND, IDC_PLAY_END, 0L) ;

            EndDialog (hwnd, 0) ;
            return TRUE ;
        }
        break ;
    }
    return FALSE ;
}
```

在研究基于消息和基于文本的 MCI 接口时，您会发现它们非常相近。很容易就可以猜测出 MCI 将命令字符串转换为相应的命令消息和数据结构。RECORD3 可以像 RECORD2 一样使用 MM_MCINOTIFY 消息，但是它没有选择 mciExecute 函数的含义，它

的缺点是程序不知道什么时候播放完波形文件，因此，这些按钮不能自动改变状态。您必须人工按下 End 按钮，以便让程序知道它已经准备好再次录音或播放。

注意 MCI 的 open 命令中 alias 关键字的用法，它允许所有后来的 MCI 命令使用别名来引用设备。

22.2.9　波形音频文件格式

如果在十六进制转储程序下研究未压缩的 .WAV 文件(即 PCM)，您会发现它们具有表 22-1 所示的格式。

这是一个被称为 RIFF(Resource Interchange File Format：资源交换文件格式)的可扩展格式的例子。RIFF 是用于多媒体数据文件的封装格式，它是一种标记文件格式，在这种格式下，文件由数据“块”组成，而这些数据块则由前面 4 个字符的 ASCII 名称和 4 字节(32 位)的块大小来确认。块大小值不包括块的名称和大小所需要的 8 字节。

表 22-1　.WAV 文件格式

偏移量	字节	数据
0000	4	“RIFF”
0004	4	波形块的大小(文件大小减 8)
0008	4	“WAVE”
000C	4	“fmt ”
0010	4	格式块的大小(16 字节)
0014	2	wf.wFormatTag = WAVE_FORMAT_PCM = 1
0016	2	wf.nChannels
0018	4	wf.nSamplesPerSec
001C	4	wf.nAvgBytesPerSec
0020	2	wf.nBlockAlign
0022	2	wf.wBitsPerSample
0024	4	“data”
0028	4	波形数据的大小
002C		波形数据

波形音频文件以文本字符串“RIFF”开始，用来标识这是一个 RIFF 文件。字符串后面是一个 32 位的块大小，表示文件其余部分的大小，或者是小于 8 字节的文件大小。

块数据以文本字符串“WAVE”开始，用来标识这是一个波形音频块，后面是文本字符串“fmt ”——注意用空白使之成为 4 字符的字符串——用来标识包含波形音频数据格式的子块。“fmt ”字符串的后面是格式信息大小，这里是 16 字节。格式信息是 WAVEFORMATEX 结构的前 16 个字节，或者像最初定义时一样，是包含 WAVEFORMAT 结构的 PCMWAVEFORMAT 结构。

nChannels 字段的值是 1 或 2,分别对应于单声道和立体声。nSamplesPerSec 字段是每秒的样本数;标准值是每秒 11 025、22 050 和 44 100 个样本。nAvgBytesPerSec 字段是采样速率,单位是每秒样本数乘以通道数,再乘以以字位为单位的每个样本的大小,然后除以 8 并向上取整。标准样本大小是 8 位和 16 位。nBlockAlign 字段是通道数乘以以字位为单位的样本大小,然后除以 8 并向上取整。最后,该格式以 wBitsPerSample 字段结束,该字段是通道数乘以以字位为单位的样本大小。

格式信息的后面是文本字符串"data",然后是 32 位的数据大小,最后是波形数据本身。这些数据是按相同格式进行简单连接的样本,这与低级波形音频设备上所使用的格式相同。如果样本大小是 8 位或者更少,那么每个样本有 1 字节用于单声道,或者有 2 字节用于立体声。如果样本大小在 9 ~ 16 位之间,则每个样本就有 2 字节用于单声道,或者 4 字节用于立体声。对于立体声波形数据,每个样本都由左值及其后面的右值组成。

对于 8 位或不到 8 位的样本大小,样本字节被解释为无符号值。例如,对于 8 位的样本大小,静音等于 0x80 字节的字符串。对于 9 位或更多的样本大小,样本被解释为有符号值,这时静音的字符串等于值 0。

用于读取标记文件的一个重要规则是忽略不准备处理的块。尽管波形音频文件需要"fmt "和"data"子块(按照此顺序),但它还包含其他子块。尤其是,波形音频文件可能包含一个标记为"INFO"的子块,以及提供波形音频文件信息的子块的子块。

22.2.10　附加合成实验

许多年来——至少可以追溯到毕达哥拉斯的年代——人们就已经试图分析音调。起初好像非常简单,但随后就变得复杂了。请原谅,我将重复一些已经说过的有关声音的问题。

音调,除了一些撞击声以外,都有特殊的音高或频率。这个频率可以在人类能够感受到的频谱范围内,也就是 20 ~ 20 000 Hz 以内。例如,钢琴的频率范围在 27.5 ~ 4 186 Hz之间。音调的另一个特征是音量或响度,这与产生音调的波形的所有振幅相应。响度的变化用分贝度量。迄今为止,一切都很好。

然后有一件难办的事称作"音质"。非常简单,音质就是声音的质量,利用它,我们可以区分按相同音调相同音量演奏的钢琴、小提琴和喇叭。

法国数学家 Fourier 发现一些周期性的波形——不论多么复杂——它们都可以表示为许多频率是基础频率整数倍的正弦波形。这个基础频率,也称作第一个谐波,是波形周期的频率。第一个泛音,也称作二级谐波,是基本频率的两倍;第二个泛音,或者三级谐波的频率是基本频率的三倍,依次类推。谐波振幅的相互关系形成了波形的形状。

例如,方波可以表示为许多的正弦波,其中偶数谐波(即 2、4、6 等)的振幅都是 0,而

奇数谐波(即 1、3、5 等)的振幅都按 1、1/3、1/5 等比例,依次类推。在锯齿波中,所有的泛音都出现,而振幅都按 1、1/2、1/3、1/4 等比例,依此类推。

对于德国科学家 Hermann Helmholtz(1821 ~ 1894),这是了解音质的关键。在他的名著《On the Sensations of Tone》(1885 年初版,1954 年由 Dover Press 再版)中,Helmholtz 假定耳朵和大脑将复杂的声音分解为正弦波,而这些正弦波相关的强度就是我们所感受的音质。遗憾的是,事情还没有这么简单。

随着 1968 年 Wendy Carlos 的唱片《Switched on Bach》的发布,电子音乐合成器引起了公众的广泛注意。那时使用的合成器(例如 Moog)是模拟合成器。这些合成器使用模拟电路来产生各种音频波形,例如方波、三角波形和锯齿波形。要使这些波形听起来更像真实的乐器,它们取决于单个音符的变化过程。波形的所有振幅按"包络(envelope)"形成。当音符开始时,振幅由 0 开始增加,通常增加得非常快,这就是所谓的冲击。然后当音符持续时,振幅保持为常数,这时称为"持续"。音符结束时,振幅降为 0,这时称为释放。

波形通过滤波器,滤波器将削弱一些谐波,并将简单波形转换得更复杂、更有乐感。这些滤波器的切断频率由包络控制,以便声音的谐波内容在音符的过程中改变。

因为这些合成器以丰富的波形格式调和开始,而且一些谐波通过滤波器进行了削弱,这种形式的合成称为"负合成"。

即使在负合成期间,许多人也还会在电子音乐中发现附加合成是下一个大问题。

在附加合成中,可以从许多接收整数倍的正弦波生成器开始,接收整数倍是为了便于每个正弦波都对应一个谐波。每个谐波的振幅都由一个包络单独控制。使用模拟电路的附加合成不实用,因为有时对某个音符就需要 8 ~ 24 之间的正弦波生成器,而与这些正弦波生成器相关的频率将正好相互跟踪。模拟波形生成器很不好,而且容易发生频率漂移。

不过,对于数字合成器(可以使用查找表数字化地产生波形)和计算机产生的波形,频率漂移并不是个问题,因而附加合成也就切实可行了。因此总的来说:在录制真实的乐曲时,可以用 Fourier 分解法将其分解成多个谐波,然后就可以确定每个谐波的相对强度,再用多个正弦波数字化地产生声音。

如果开始实验时用 Fourier 分析法分析实际的音调,并从多个正弦波来产生这些音调,那么人们将发现音质并不像 Helmholtz 所认为的那样简单。

最大的问题是真实音调的谐波之间并没有精确的整数关系。事实上,"谐波"一词对于实际的音调来说并不十分适当。各种正弦波组成都不和谐,或者更准确地说是"泛音"。

人们发现,实际音调泛音之间的不和谐在创造"真实的"声音时很重要。静态和谐会产生"电流"声。每个泛音都在单个音符上改变振幅和频率。泛音中,相对频率和振幅的关系对于不同的泛音以及来自相同乐器的不同强度是不同的。实际音调中最复杂

的部分发生在音符的冲击部分，这时比较不和谐。人们发现音符的这个复杂的冲击位置对于人类感受音质很重要。

简而言之，实际乐器的声音比任何想象的都更复杂。分析音调的观点，以及后面用于控制泛音的振幅和频率的相对简单的包络观点显然都不实用。

实际乐曲的一些分析法发表于早期（1977 和 1978 年）的《Computer Music Journal》（当时由 People's Computer Company 出版，现在由 MIT Press 出版）。由 James A. Moorer、John Grey 和 John Strawn Some 编写了第三部分丛书《Lexicon of Analyzed Tones》，该书显示了在小提琴、双簧管、单簧管和喇叭上演奏一个音符（小于半秒种）的泛音的振幅和频率图形。所用的音符是中音 C 上的降 E。小提琴用 20 个泛音，双簧管和单簧管用 21 个，而喇叭用 12 个。实际上，《Computer Music Journal》的 Volume II、Number 2（1978 年 9 月）包含了用线段来近似双簧管、单簧管和喇叭的不同频率和振幅的包络。

因此，利用 Windows 上支持的波形，下面的过程很简单：将这些数字键入程序、为每个泛音都产生多个正弦波样本、添加这些样本并将其发送给波形音频声卡，因此把 20 年前原始录制的声音重新制造出来也很容易。ADDSYNTH（"附加合成"）程序如程序 22-6 所示。

程序 22-6　ADDSYNTH 程序

ADDSYNTH.C

```
/*-----------------------------
   ADDSYNTH.C——Additive Synthesis Sound Generation
               (c) Charles Petzold, 1998
   -----------------------------*/

#include <windows.h>
#include <math.h>
#include "addsynth.h"
#include "resource.h"

#define ID_TIMER            1
#define SAMPLE_RATE         22050
#define MAX_PARTIALS        21
#define PI                  3.14159

BOOL CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName [] = TEXT ("AddSynth") ;

// Sine wave generator
// ---------
```

```
double SineGenerator (double dFreq, double * pdAngle)
{
     double dAmp ;

     dAmp = sin ( * pdAngle) ;
     * pdAngle += 2 * PI * dFreq / SAMPLE_RATE ;

     if ( * pdAngle >= 2 * PI)
        * pdAngle -= 2 * PI ;

     return dAmp ;
}

// Fill a buffer with composite waveform
// - - - - - - - - - - - - - - - -

VOID FillBuffer (INS ins, PBYTE pBuffer, int iNumSamples)
{
     static double dAngle [MAX_PARTIALS] ;
     double        dAmp, dFrq, dComp, dFrac ;
     int           i, iPrt, iMsecTime, iCompMaxAmp, iMaxAmp, iSmp ;

          // Calculate the composite maximum amplitude

     iCompMaxAmp = 0 ;

     for (iPrt = 0 ; iPrt < ins.iNumPartials ; iPrt++)
     {
          iMaxAmp = 0 ;

          for (i = 0 ; i < ins.pprt[iPrt].iNumAmp ; i++)
              iMaxAmp = max (iMaxAmp, ins.pprt[iPrt].pEnvAmp[i].iValue) ;

          iCompMaxAmp += iMaxAmp ;
     }

          // Loop through each sample

     for (iSmp = 0 ; iSmp < iNumSamples ; iSmp++)
     {
          dComp = 0 ;
          iMsecTime = (int) (1000 * iSmp / SAMPLE_RATE) ;

               // Loop through each partial

          for (iPrt = 0 ; iPrt < ins.iNumPartials ; iPrt++)
          {
               dAmp = 0 ;
```

```
                dFrq = 0 ;

                for (i = 0 ; i < ins.pprt[iPrt].iNumAmp - 1 ; i++)
                {
                    if (iMsecTime >= ins.pprt[iPrt].pEnvAmp[i].iTime &&
                        iMsecTime <= ins.pprt[iPrt].pEnvAmp[i+1].iTime)
                    {
                        dFrac = (double) (iMsecTime -
                                ins.pprt[iPrt].pEnvAmp[i].iTime) /
                                (ins.pprt[iPrt].pEnvAmp[i+1].iTime -
                                ins.pprt[iPrt].pEnvAmp[i].iTime) ;

                        dAmp = dFrac * ins.pprt[iPrt].pEnvAmp[i+1].iValue +
                            (1 - dFrac) * ins.pprt[iPrt].pEnvAmp[i].iValue ;

                        break ;
                    }
                }

                for (i = 0 ; i < ins.pprt[iPrt].iNumFrq - 1 ; i++)
                {
                    if (iMsecTime >= ins.pprt[iPrt].pEnvFrq[i].iTime &&
                        iMsecTime <= ins.pprt[iPrt].pEnvFrq[i+1].iTime)
                    {
                        dFrac = (double) (iMsecTime -
                                ins.pprt[iPrt].pEnvFrq[i].iTime) /
                                (ins.pprt[iPrt].pEnvFrq[i+1].iTime -
                                ins.pprt[iPrt].pEnvFrq[i].iTime) ;

                        dFrq = dFrac * ins.pprt[iPrt].pEnvFrq[i+1].iValue +
                                (1 - dFrac) * ins.pprt[iPrt].pEnvFrq[i].iValue ;

                        break ;
                    }
                }
                dComp += dAmp * SineGenerator (dFrq, dAngle + iPrt) ;
            }
            pBuffer[iSmp] = (BYTE) (127 + 127 * dComp / iCompMaxAmp) ;
        }
    }

// Make a waveform file
// - - - - - - - - - -

BOOL MakeWaveFile (INS ins, TCHAR * szFileName)
{
    DWORD           dwWritten ;
    HANDLE          hFile ;
    int             iChunkSize, iPcmSize, iNumSamples ;
```

```
    PBYTE           pBuffer ;
    WAVEFORMATEX    waveform ;

    hFile = CreateFile (szFileName, GENERIC_WRITE, 0, NULL,
                        CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL) ;

    if (hFile == NULL)
        return FALSE ;

    iNumSamples = ((long) ins.iMsecTime * SAMPLE_RATE / 1000 + 1) / 2 * 2 ;
    iPcmSize    = sizeof (PCMWAVEFORMAT) ;
    iChunkSize  = 12 + iPcmSize + 8 + iNumSamples ;

    if (NULL == (pBuffer = malloc (iNumSamples)))
    {
        CloseHandle (hFile) ;
        return FALSE ;
    }

    FillBuffer (ins, pBuffer, iNumSamples) ;

    waveform.wFormatTag         = WAVE_FORMAT_PCM ;
    waveform.nChannels          = 1 ;
    waveform.nSamplesPerSec     = SAMPLE_RATE ;
    waveform.nAvgBytesPerSec    = SAMPLE_RATE ;
    waveform.nBlockAlign        = 1 ;
    waveform.wBitsPerSample     = 8 ;
    waveform.cbSize             = 0 ;

    WriteFile (hFile, "RIFF",          4, &dwWritten, NULL) ;
    WriteFile (hFile, &iChunkSize,     4, &dwWritten, NULL) ;
    WriteFile (hFile, "WAVEfmt ",      8, &dwWritten, NULL) ;
    WriteFile (hFile, &iPcmSize,       4, &dwWritten, NULL) ;
    WriteFile (hFile, &waveform, sizeof (WAVEFORMATEX) - 2, &dwWritten, NULL) ;
    WriteFile (hFile, "data",          4, &dwWritten, NULL) ;
    WriteFile (hFile, &iNumSamples,    4, &dwWritten, NULL) ;
    WriteFile (hFile, pBuffer,         iNumSamples, &dwWritten, NULL) ;

    CloseHandle (hFile) ;
    free (pBuffer) ;

    if ((int) dwWritten != iNumSamples)
    {
        DeleteFile (szFileName) ;
        return FALSE ;
    }
    return TRUE ;
}
```

```
void TestAndCreateFile (HWND hwnd, INS ins, TCHAR * szFileName, int idButton)
{
     TCHAR szMessage [64] ;

     if ( -1 != GetFileAttributes (szFileName))
          EnableWindow (GetDlgItem (hwnd, idButton), TRUE) ;
     else
     {
          if (MakeWaveFile (ins, szFileName))
               EnableWindow (GetDlgItem (hwnd, idButton), TRUE) ;
          else
          {
               wsprintf (szMessage, TEXT ("Could not create %x."), szFileName) ;
               MessageBeep (MB_ICONEXCLAMATION) ;
               MessageBox (hwnd, szMessage, szAppName,
                                   MB_OK | MB_ICONEXCLAMATION) ;
          }
     }
}

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     if ( -1 == DialogBox (hInstance, szAppName, NULL, DlgProc))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
     }
     return 0 ;
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static TCHAR * szTrum = TEXT ("Trumpet.wav") ;
     static TCHAR * szOboe = TEXT ("Oboe.wav") ;
     static TCHAR * szClar = TEXT ("Clarinet.wav") ;

     switch (message)
     {
     case WM_INITDIALOG:
          SetTimer (hwnd, ID_TIMER, 1, NULL) ;
          return TRUE ;

     case WM_TIMER:
          KillTimer (hwnd, ID_TIMER) ;
          SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
```

```
            ShowCursor (TRUE) ;

            TestAndCreateFile (hwnd, insTrum, szTrum, IDC_TRUMPET) ;
            TestAndCreateFile (hwnd, insOboe, szOboe, IDC_OBOE) ;
            TestAndCreateFile (hwnd, insClar, szClar, IDC_CLARINET) ;

            SetDlgItemText (hwnd, IDC_TEXT, TEXT ("")) ;
            SetFocus (GetDlgItem (hwnd, IDC_TRUMPET)) ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;
            return TRUE ;

        case WM_COMMAND:
            switch (LOWORD (wParam))
            {
            case IDC_TRUMPET:
                PlaySound (szTrum, NULL, SND_FILENAME | SND_SYNC) ;
                return TRUE ;

            case IDC_OBOE:
                PlaySound (szOboe, NULL, SND_FILENAME | SND_SYNC) ;
                return TRUE ;

            case IDC_CLARINET:
                PlaySound (szClar, NULL, SND_FILENAME |SND_SYNC) ;
                return TRUE ;
            }
            break ;

        case WM_SYSCOMMAND:
            switch (LOWORD (wParam))
            {
            case SC_CLOSE:
                EndDialog (hwnd, 0) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}
```

ADDSYNTH.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

# include "resource.h"
```

```
# include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog

ADDSYNTH DIALOG DISCARDABLE 100, 100, 176, 49
STYLE WS_MINIMIZEBOX | WS_CAPTION | WS_SYSMENU
CAPTION "Additive Synthesis"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON      "Trumpet",IDC_TRUMPET,8,8,48,16
    PUSHBUTTON      "Oboe",IDC_OBOE,64,8,48,16
    PUSHBUTTON      "Clarinet",IDC_CLARINET,120,8,48,16
    LTEXT           "Preparing Data...",IDC_TEXT,8,32,100,8
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by AddSynth.rc

# define IDC_TRUMPET            1000
# define IDC_OBOE               1001
# define IDC_CLARINET           1002
# define IDC_TEXT               1003
```

这里没有给出附加文件 ADDSYNTH.H，因为它包含几百行令人讨厌的语句，您可以在本书附带的盘上找到它。在 ADDSYNTH.H 的开始位置，我定义了三个结构，用于存储包络数据。每个振幅和频率分别存储到类型 ENV 的结构数组中。这些数字对由时间（毫秒）和振幅值（按任意度量单位）或频率（以 Hz 为单位）组成。这些数组的长度可变，其变化范围为 6～14。假定振幅和频率值之间是直线连接。

每种乐器都包括一个泛音集（喇叭用 12 个，双簧管和单簧管分别使用 21 个），这些泛音集保存在类型 PRT 的结构数组中。PRT 结构保存振幅和频率包络的点数，以及指向 ENV 数组的指针。INS 结构包括音调的总时间（以毫秒为单位）、泛音数以及指向保存泛音的 PRT 数组的指针。

ADDSYNTH 有三个标记为 Trumpet、Oboe 和 Clarinet 的按钮。PC 的速度还没有快到足以实时计算所有的附加合成，因此第一次运行 ADDSYNTH 时，这些按钮将失效，直到程序计算完样本并创建了 TRUMPET.WAV、OBOE.WAV 和 CLARINET.WAV 声音文件后，按钮才被激活，而且可以使用 PlaySound 函数播放这三种声音。下次运行时，程序将检查波形文件是否存在，而不需重新创建。

ADDSYNTH 中的 FillBuffer 函数完成了大多数工作。FillBuffer 从计算合成最大振幅的总数开始。为此，它在乐器的泛音中循环，以找出每个泛音的最大振幅，然后将所有

的最大振幅加起来。此值后来用于将样本缩放到 8 位的样本大小。

然后 FillBuffer 计算每个样本的值。每个样本都对应于一段毫秒时间，该时间取决于采样速率。（实际上，在 22.05 kHz 的采样速率下，每 22 个样本对应于相同的毫秒时间值。）然后，FillBuffer 在泛音中循环。对于频率和振幅，它找出与毫秒时间值对应的包络线段，并执行线形插补。

频率值与相位角值一起传递给 SineGenerator 函数。本章前面讨论过，产生数字化的正弦波形需要保持相位角值，并基于频率值增加。从 SineGenerator 函数返回时，正弦值将乘以泛音的振幅并累加。样本的所有泛音都加在一起之后，样本就缩放到字节大小。

22.2.11　唤醒波形音频

WAKEUP，如程序 22-7 所示，是源代码文件看起来不很完整的程序之一。程序窗口看起来像对话框，但是没有资源描述文件（我们已经知道如何编写），并且程序使用一个波形文件，但在光盘上却没有这样的文件。不过，程序非常有趣：它播放的声音很大，但非常令人讨厌。WAKEUP 是我的闹钟，能够无休止地为唤醒我而工作。

程序 22-7　WAKEUP 程序

WAKEUP.C

```
/*------------------------
   WAKEUP.C——Alarm Clock Program
            (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>
#include <commctrl.h>

     // ID values for 3 child windows

#define ID_TIMEPICK  0
#define ID_CHECKBOX  1
#define ID_PUSHBTN   2

     // Timer ID

#define ID_TIMER        1

     // Number of 100-nanosecond increments (ie FILETIME ticks) in an hour
#define FTTICKSPERHOUR (60 * 60 * (LONGLONG) 10000000)
```

```
        // Defines and structure for waveform "file"

#define SAMPRATE    11025
#define NUMSAMPS    (3 * SAMPRATE)
#define HALFSAMPS   (NUMSAMPS / 2)

typedef struct
{
    char            chRiff[4] ;
    DWORD           dwRiffSize ;
    char            chWave[4] ;
    char            chFmt [4] ;
    DWORD           dwFmtSize ;
    PCMWAVEFORMAT   pwf ;
    char            chData[4] ;
    DWORD           dwDataSize ;
    BYTE            byData[0] ;
}
WAVEFORM ;

        // The window proc and the subclass proc

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
LRESULT CALLBACK SubProc (HWND, UINT, WPARAM, LPARAM) ;

        // Original window procedure addresses for the subclassed windows

WNDPROC SubbedProc [3] ;

        // The current child window with the input focus

HWND hwndFocus ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInst,
                    PSTR szCmdLine, int iCmdShow)
{
    static TCHAR szAppName [] = TEXT ("WakeUp") ;
    HWND         hwnd ;
    MSG          msg ;
    WNDCLASS     wndclass ;

    wndclass.style          = 0 ;
    wndclass.lpfnWndProc    = WndProc ;
    wndclass.cbClsExtra     = 0 ;
    wndclass.cbWndExtra     = 0 ;
    wndclass.hInstance      = hInstance ;
    wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
```

```
     wndclass.hbrBackground  = (HBRUSH) (1 + COLOR_BTNFACE) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, szAppName,
                          WS_OVERLAPPED | WS_CAPTION |
                                          WS_SYSMENU | WS_MINIMIZEBOX,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static HWND                hwndDTP, hwndCheck, hwndPush ;
     static WAVEFORM            waveform = { "RIFF", NUMSAMPS + 0x24, "WAVE", "fmt ",
                                             sizeof (PCMWAVEFORMAT), 1, 1, SAMPRATE,
                                             SAMPRATE, 1, 8, "data", NUMSAMPS } ;
     static WAVEFORM          * pwaveform ;
     FILETIME                   ft ;
     HINSTANCE                  hInstance ;
     INITCOMMONCONTROLSEX       icex ;
     int                        i, cxChar, cyChar ;
     LARGE_INTEGER              li ;
     SYSTEMTIME                 st ;

     switch (message)
     {
     case WM_CREATE:
          // Some initialization stuff
```

```
hInstance = (HINSTANCE) GetWindowLong (hwnd, GWL_HINSTANCE) ;

icex.dwSize = sizeof (icex) ;
icex.dwICC  = ICC_DATE_CLASSES ;
InitCommonControlsEx (&icex) ;

    // Create the waveform file with alternating square waves

pwaveform = malloc (sizeof (WAVEFORM) + NUMSAMPS) ;
* pwaveform = waveform ;

for (i = 0 ; i < HALFSAMPS ; i++)
    if (i % 600 < 300)
        if (i % 16 < 8)
            pwaveform -> byData[i] = 25 ;
        else
            pwaveform -> byData[i] = 230 ;
    else
        if (i % 8 < 4)
            pwaveform -> byData[i] = 25 ;
        else
            pwaveform -> byData[i] = 230 ;

    // Get character size and set a fixed window size.

cxChar = LOWORD (GetDialogBaseUnits ()) ;
cyChar = HIWORD (GetDialogBaseUnits ()) ;

SetWindowPos (hwnd, NULL, 0, 0,
              42 * cxChar,
              10 * cyChar / 3 + 2 * GetSystemMetrics (SM_CYBORDER) +
                                    GetSystemMetrics (SM_CYCAPTION),
              SWP_NOMOVE | SWP_NOZORDER | SWP_NOACTIVATE) ;

    // Create the three child windows

hwndDTP = CreateWindow (DATETIMEPICK_CLASS, TEXT (""),
              WS_BORDER | WS_CHILD | WS_VISIBLE | DTS_TIMEFORMAT,
              2 * cxChar, cyChar, 12 * cxChar, 4 * cyChar / 3,
              hwnd, (HMENU) ID_TIMEPICK, hInstance, NULL) ;
hwndCheck = CreateWindow (TEXT ("Button"), TEXT ("Set Alarm"),
              WS_CHILD | WS_VISIBLE | BS_AUTOCHECKBOX,
              16 * cxChar, cyChar, 12 * cxChar, 4 * cyChar / 3,
              hwnd, (HMENU) ID_CHECKBOX, hInstance, NULL) ;

hwndPush = CreateWindow (TEXT ("Button"), TEXT ("Turn Off"),
              WS_CHILD | WS_VISIBLE | BS_PUSHBUTTON | WS_DISABLED,
```

```
                    28 * cxChar, cyChar, 12 * cxChar, 4 * cyChar / 3,
                    hwnd, (HMENU) ID_PUSHBTN, hInstance, NULL) ;

    hwndFocus = hwndDTP ;

        // Subclass the three child windows

    SubbedProc [ID_TIMEPICK] = (WNDPROC)
                SetWindowLong (hwndDTP, GWL_WNDPROC, (LONG) SubProc) ;
    SubbedProc [ID_CHECKBOX] = (WNDPROC)
                SetWindowLong (hwndCheck, GWL_WNDPROC, (LONG) SubProc);
    SubbedProc [ID_PUSHBTN] = (WNDPROC)
                SetWindowLong (hwndPush, GWL_WNDPROC, (LONG) SubProc) ;

        // Set the date and time picker control to the current time
        // plus 9 hours, rounded down to next lowest hour

    GetLocalTime (&st) ;
    SystemTimeToFileTime (&st, &ft) ;
    li = * (LARGE_INTEGER *) &ft ;
    li.QuadPart += 9 * FTTICKSPERHOUR ;
    ft = * (FILETIME *) &li ;
    FileTimeToSystemTime (&ft, &st) ;
    st.wMinute = st.wSecond = st.wMilliseconds = 0 ;
    SendMessage (hwndDTP, DTM_SETSYSTEMTIME, 0, (LPARAM) &st) ;
    return 0 ;

case WM_SETFOCUS:
    SetFocus (hwndFocus) ;
    return 0 ;

case WM_COMMAND:
    switch (LOWORD (wParam))          // control ID
    {
    case ID_CHECKBOX:

            // When the user checks the "Set Alarm" button, get the
            // time in the date and time control and subtract from
            // it the current PC time.

        if (SendMessage (hwndCheck, BM_GETCHECK, 0, 0))
        {
            SendMessage (hwndDTP, DTM_GETSYSTEMTIME, 0, (LPARAM) &st) ;
            SystemTimeToFileTime (&st, &ft) ;
            li = * (LARGE_INTEGER *) &ft ;

            GetLocalTime (&st) ;
```

```
                SystemTimeToFileTime (&st, &ft) ;
                li.QuadPart -= ((LARGE_INTEGER *) &ft) -> QuadPart ;

                    // Make sure the time is between 0 and 24 hours!
                    // These little adjustments let us completely ignore
                    // the date part of the SYSTEMTIME structures.

                while (li.QuadPart < 0)
                      li.QuadPart += 24 * FTTICKSPERHOUR ;

                li.QuadPart %= 24 * FTTICKSPERHOUR ;

                    // Set a one-shot timer! (See you in the morning.)

                SetTimer (hwnd, ID_TIMER, (int) (li.QuadPart / 10000), 0) ;
            }
            // If button is being unchecked, kill the timer.

            else
                KillTimer (hwnd, ID_TIMER) ;

            return 0 ;

            // The "Turn Off" button turns off the ringing alarm, and also
            // unchecks the "Set Alarm" button and disables itself.

        case ID_PUSHBTN:
            PlaySound (NULL, NULL, 0) ;
            SendMessage (hwndCheck, BM_SETCHECK, 0, 0) ;
            EnableWindow (hwndDTP, TRUE) ;
            EnableWindow (hwndCheck, TRUE) ;
            EnableWindow (hwndPush, FALSE) ;
            SetFocus (hwndDTP) ;
            return 0 ;
        }
        return 0 ;
            // The WM_NOTIFY message comes from the date and time picker.
            // If the user has checked "Set Alarm" and then gone back to
            // change the alarm time, there might be a discrepancy between
            // the displayed time and the one-shot timer. So the program
            // unchecks "Set Alarm" and kills any outstanding timer.

    case WM_NOTIFY:
        switch (wParam)             // control ID
        {
        case ID_TIMEPICK:
            switch (((NMHDR *) lParam) -> code)         // notification code
```

```
            }
            case DTN_DATETIMECHANGE:
                if (SendMessage (hwndCheck, BM_GETCHECK, 0, 0))
                {
                    KillTimer (hwnd, ID_TIMER) ;
                    SendMessage (hwndCheck, BM_SETCHECK, 0, 0) ;
                }
                return 0 ;
            }
        }
        return 0 ;

        // The WM_COMMAND message comes from the two buttons.

    case WM_TIMER:

            // When the timer message comes, kill the timer (because we only
            // want a one-shot) and start the annoying alarm noise going.
        KillTimer (hwnd, ID_TIMER) ;
        PlaySound ((PTSTR) pwaveform, NULL,
                   SND_MEMORY | SND_LOOP | SND_ASYNC);

            // Let the sleepy user turn off the timer by slapping the
            // space bar. If the window is minimized, it's restored; then
            // it's brought to the forefront; then the pushbutton is enabled
            // and given the input focus.

        EnableWindow (hwndDTP, FALSE) ;
        EnableWindow (hwndCheck, FALSE) ;
        EnableWindow (hwndPush, TRUE) ;

        hwndFocus = hwndPush ;
        ShowWindow (hwnd, SW_RESTORE) ;
        SetForegroundWindow (hwnd) ;
        return 0 ;

        // Clean up if the alarm is ringing or the timer is still set.

    case WM_DESTROY:
        free (pwaveform) ;

        if (IsWindowEnabled (hwndPush))
            PlaySound (NULL, NULL, 0) ;

        if (SendMessage (hwndCheck, BM_GETCHECK, 0, 0))
            KillTimer (hwnd, ID_TIMER) ;
```

```
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

LRESULT CALLBACK SubProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    int idNext, id = GetWindowLong (hwnd, GWL_ID) ;

    switch (message)
    {
    case WM_CHAR:
        if (wParam == '\t')
        {
            idNext = id ;

            do
                idNext = (idNext +
                    (GetKeyState (VK_SHIFT) < 0 ? 2 : 1)) % 3 ;
            while (!IsWindowEnabled (GetDlgItem (GetParent (hwnd), idNext))) ;

            SetFocus (GetDlgItem (GetParent (hwnd), idNext)) ;
            return 0 ;
        }
        break ;

    case WM_SETFOCUS:
        hwndFocus = hwnd ;
        break ;
    }
    return CallWindowProc (SubbedProc [id], hwnd, message, wParam, lParam) ;
}
```

WAKEUP 使用的波形只有两个方波，但是它们快速变化。实际的波形在 WndProc 的 WM_CREATE 消息期间计算。所有的波形文件都存储在内存中，指向这个内存块的指针传递给 PlaySound 函数，该函数使用 SND_MEMORY、SND_LOOP 和 SND_ASYNC 参数。

WAKEUP 使用被称为“Date-Time Picker”的通用控件。这个控件用逻辑来允许用户选择特殊的日期和时间。（WAKEUP 只使用时间特征。）程序可以使用 SYSTEMTIME 结构来获得和设置时间，在获得和设置 PC 自身时钟时也使用该结构。要多方面了解 Date-Time Picker，请试着创建不带有任何 DTS 类型标记的窗口。

注意 WM_CREATE 消息结束时的逻辑：程序假定您在睡觉之前运行它，并希望它在 8 小时之后来唤醒您。

现在很明显，可以从 GetLocalTime 函数在 SYSTEMTIME 结构获得当前时间，而且可以“手工”增加时间。但在一般情况下，此计算将涉及检查大于 24 小时的结果时间，这意味着您必须增加天数字段，然后可能涉及增加月（因此还必须有用于每月天数和闰年检查的逻辑），最后您可能还要增加年。

事实上，推荐的方法（来自/Platform SDK/Windows Base Services/General Library/Time /Time Reference/Time Structures/SYSTEMTIME）是将 SYSTEMTIME 转换为 FILETIME 结构（使用 SystemTimeToFileTime），将 FILETIME 结构强制转换为 LARGE_INTEGER 结构，在大整数上执行计算，再强制转换回 FILETIME 结构，然后转换回 SYSTEMTIME 结构（使用 FileTimeToSystemTime）。

顾名思义，FILETIME 结构用于获得和设置文件最后一次更新的时间。此结构如下：

```
type struct_FILETIME          // ft
{
    DWORD dwLowDateTime ;
    DWORD dwHighDateTime ;
}
FILETIME ;
```

这两个字段一起表示了从 1601 年 1 月 1 日起每隔 100×10^{-9} 秒所显示的 64 位值。

Microsoft C/C++ 编译器支持 64 位整数作为 ANSI C 的非标准扩展。其数据类型是 __int64。您可以对 __int64 类型执行所有的常规算术运算，并且有一些运行库函数也支持它们。Windows 的 WINNT.H 头文件定义如下：

```
typedef __int64 LONGLONG ;
typedef unsigned __int64 DWORDLONG ;
```

在 Windows 中，这有时称为“4 字”，或者更普遍地称为“大整数”。还有一个联合定义如下：

```
typedef union _LARGE_INTEGER
{
    struct
    {
        DWORD LowPart ;
        LONG HighPart ;
    } ;
    LONGLONG QuadPart ;
}
LARGE_INTEGER ;
```

这是/Platform SDK/Windows Base Services/General Library/Large Integer Operations 中的全部文档。此联合允许您使用 32 位或者 64 位的大整数。

22.3 MIDI 和音乐

由电子音乐合成器制造者协会在 20 世纪 80 年代早期开发了“乐器数字化接口(MIDI: Musical Instrument Digital Interface)”。MIDI 是用于将它们中的电子乐器与计算机连接起来的协议,也是电子音乐领域中相当重要的标准。MIDI 规范由 MIDI Manufacturers Association(MMA)维护,它的 Web 站点是 http://www.midi.org。

22.3.1 使用 MIDI

MIDI 为通过电缆来传递数字命令定义协议。MIDI 电缆使用 5 针 DIN 连接器,但是只使用了三个连接器:一个是屏蔽,一个是当前循环,而第三个传输数据。MIDI 协议在每秒 31 250 位的速度下是单向的。数据的每个字节都由一个开始位开始,以一个停止位结束,用于每秒 3125 字节的有效传输速率。

重要的是要认识到没有真实的声音——不论是模拟格式还是数字格式——通过 MIDI 电缆传输。通过电缆传输的通常都是简单的消息,长度一般是 1、2 或 3 字节。

简单的 MIDI 配置可以包括两片 MIDI 兼容硬件。一个是本身不发声,但是单独产生 MIDI 消息的 MIDI 键盘。此键盘有一个有标记有“MIDI Out”的 MIDI 端口。用 MIDI 电缆将这个端口与 MIDI 声音合成器的“MIDI In”端口连接起来。合成器看起来很像前面有几个按钮的小盒子。

按下键盘上的一个键时(假定是中音 C),键盘就将 3 个字节发送给 MIDI Out 端口。在十六进制中,这些字节是:

```
90 3C 40
```

第 1 个字节(90)显示 Note On 消息。第 2 个字节是键号,其中 3C 是中音 C。第 3 个字节是敲击键的速度,此速度范围是 1 ~ 127。我们恰巧使用了一个对速度不敏感的键盘,因此它发送平均速度值。这个 3 字节的消息顺着 MIDI 电缆进入合成器的 Midi In 端口。通过播放中音 C 的音调来响应合成器。

释放键时,键盘会将另一个 3 字节消息发送给 MIDI Out 端口:

```
90 3C 00
```

这与 Note On 命令相同,但带有 0 速字节。这个 0 字节表示 Note Off 命令,意味着应该关闭音符。合成器通过停止声音来响应。

如果合成器有复调音乐的能力(即同时播放多个音符的能力),那么您就可以在键盘上演奏和弦。键盘产生多个 Note On 消息,并且合成器将播放所有的音符。当您释放和弦时,键盘就将多个 Note Off 消息发送给合成器。

一般来说,这种配置中的键盘称为“MIDI 控制器”,它负责产生 MIDI 消息来控制合

成器。MIDI 控制器看起来不像键盘。MIDI 控制器包括下面几种：看起来像单簧管或萨克斯管的 MID 管乐控制器、MIDI 吉他控制器、MIDI 弦乐控制器和 MIDI 鼓控制器。至少所有这些控制器都产生 3 字节的 Note On 和 Note Off 消息。

胜过类似的键盘或传统乐器，控制器也可以是“音序器”，它是在内存中存储 Note On 和 Note Off 消息顺序，然后再播放的硬件。今天使用单机音序器的已经比几年前少多了，因为它们已经被计算机所替代。安装 MIDI 卡的计算机也生成 Note On 和 Note Off 消息来控制合成器。MIDI 编辑软件，允许您在屏幕上作曲，还可以存储来自 MIDI 控制器的 MIDI 消息，并处理这些消息，然后将 MIDI 消息发送给合成器。

合成器有时也称为“声音模块(sound module)”或“音调生成器(tone generator)”。MIDI 不指定真正产生这些声音的方法。合成器可以使用任何一种声音生成技术。

实际上，只有非常简单的 MIDI 控制器(例如管乐控制器)才会有 MIDI Out 电缆端口。通常键盘都有内置的合成器，并且有三个 MIDI 电缆端口，分别标记为“MIDI In”、“MIDI Out”和“MIDI Thru”。MIDI In 端口接受 MIDI 消息，以播放键盘的内部合成器。MIDI Out 端口将 MIDI 消息从键盘发送到外部合成器。MIDI Thru 端口是一个输出端口，它复制 MIDI In 端口的输入信号——无论从 MIDI In 端口获得什么都发送给 MIDI Thru 端口。(MIDI Thru 端口不包括从 MIDI Out 端口发送的任何信息。)

通过电缆连接 MIDI 硬件只有两种方法：将一个硬件上的 MIDI Out 连接到另一个的 MIDI In，或者将 MIDI Thru 与 MIDI In 连接。MIDI Thru 端口允许连接一流的 MIDI 合成器。

22.3.2 更改程序

合成器能制作哪种声音？是钢琴声、小提琴声、喇叭声，还是飞碟声？通常合成器能够生成的各种声音都保存在 ROM 或者其他地方，它们通常被称为“声音”、“乐器”或者“音色”。(“音色”一词来自模拟合成器的时代，当时通过将音色和弦插入合成器前面的插孔中来配置不同的声音。)

在 MIDI 中，合成器能够生成的各种声音被称为“程序”。更改这个程序需要向合成器发送 MIDI Program Change 消息：

```
C0 pp
```

其中，pp 的范围是 0～127。通常 MIDI 键盘的顶部是一系列有限的按钮，这些按钮将产生 Program Change 消息。通过按下这些按钮，您可以从键盘控制合成器的声音。这些按钮号通常由 1 开始，而不是由 0 开始，因此程序号 1 与 Program Change 字节的 0 对应。

MIDI 规范没有说明程序号与乐器的对应关系。例如，著名的 Yamaha DX7 合成器上的前三个程序分别称为“Warm Strings”、“ Mellow Horn”和“Pick Guitar”。而在 Yamaha TX81Z 音调发生器上，它们是 Grand Piano、Upright Piano 和 Deep Grand。在 Roland MT-32

声音模块上，它们是 Acoustic Piano 1、Acoustic Piano 2 和 Acoustic Piano 3。因此，如果您不想对通过敲击键盘使程序改变感到吃惊，那么最好了解一下乐器声与您将使用的合成器的程序号的对应关系。

这对于包含 Program Change 消息的 MIDI 文件来说是一个实际问题——这些文件并不是设备无关的，因为它们的内容在不同的合成器上听起来是不一样的。然而，在最近几年，“General MIDI(GM)”标准已经把这些程序号标准化。Windows 支持 General MIDI。如果合成器与 General MIDI 规范不一致，那么程序映射可使它仿真 General MIDI 合成器。

22.3.3 MIDI 通道

迄今为止，我已经讨论了两个 MIDI 消息，第一个是 Note On，

```
90 kk vv
```

其中，kk 是键号(0 ~ 127)，vv 是速度(0 ~ 127)。0 速度表示 Note Off 命令。第二个是 Program Change，

```
C0 pp
```

其中，pp 的范围是 0 ~ 127。这些是典型的 MIDI 消息。第一个字节称作“状态”字节。根据字节的状态，它通常后跟 0、1 或 2 字节的“数据”。(我即刻将说明的“系统专有”消息除外。)从数据字节中分辨出状态字节容易：高位总是 1 用于状态字节，0 用于数据字节。

然而，我还没有讨论过这两个消息的普通格式。Note On 消息的普通格式如下：

```
9n kk vv
```

而 Program Change 是

```
Cn pp
```

在这两种情况下，n 表示状态字节的低 4 位，其变化范围是 0 ~ 15。这就是 MIDI“通道”。通道一般从 1 开始编号，因此，如果 n 为 0，则代表通道 1。

使用 16 个不同通道允许一条 MIDI 电缆传输 16 种不同声音的消息。通常，您将发现 MIDI 消息的特殊字符串以 Program Change 消息开始，为所用的不同通道设置声音，而字符串的后面是多个 Note On 和 Note Off 命令，再后面可能是其他的 Program Change 命令，但任何时候，每个通道都只与一种声音联系。

让我们看一个简单例子：假定已经讨论过的键盘控件能够同时产生用于两条不同通道——通道 1 和通道 2——的 MIDI 消息。通过按下键盘上的按钮将两个 Program Change 消息发送给合成器：

```
C0 01
C1 05
```

现在设置通道 1 用于程序 2,并设置通道 2 用于程序 6。(您或许还回忆得出通道号和程序号都是基于 1 的,但消息中的编码是基于 0 的。)现在按下键盘上的键时,就发送两个 Note On 消息,每个通道一个:

```
90 kk vv
91 kk vv
```

这就允许您和谐地同时播放两种乐器的声音。

另一种方法是"分开"键盘。低键可以在通道 1 上产生 Note On 消息,高键可以在通道 2 上产生 Note On 消息。这就允许您在一个键盘上独立播放两种乐器的声音。

当您考虑 PC 上的 MIDI 序列软件时,使用 16 个通道将更为有利。每个通道都代表不同的乐器。如果有能够独立播放 16 种不同乐器的合成器,那么就可以编写用于 16 个波段的管弦乐曲,而且只使用一条 MIDI 电缆将 MIDI 卡与合成器连接起来。

22.3.4 MIDI 消息

尽管 Note On 和 Program Change 消息在任何 MIDI 执行中都是最重要的消息,但并不是所有的 MIDI 都可以执行。下页表 22-2 是 MIDI 规范中定义的 MIDI 通道消息表。在前面提到过,状态字节总是高位设置,而状态字节后面的数据字节的高位都等于 0。这意味着状态字节的范围是 0x80 ~ 0xFF,而数据字节的范围是 0 ~ 0x7F。

尽管它们不必这样,但是键号通常还是与西方音乐的传统音符相对应。(例如,对于打击声音,每个键号码可以是不同的打击乐器。)当键号与钢琴类的键盘对应时,键 60(十进制)是中音 C。MIDI 键号在普通的 88 键钢琴范围的基础上向下扩展 21 个音符,向上扩展 19 个音符。速度号是按下某键的速度,在钢琴上它控制声音的响度与和谐特征。特殊的声音可以按这种方式或其他方式来响应键的速度。

前面演示的例子使用带有 0 速字节的 Note On 消息来表示 Note Off 命令。对于键盘(或者其他控制器)还有一个单独的 Note Off 命令,该命令实现释放键的速度,不过非常少见。

还有两个"接触后"消息。接触后是一些键盘的特征,按下某个键以后,再用力按下键可以在某些方式上改变声音。一个消息(状态字节 0xDn)是将接触后应用于通道中当前演奏的所有音符,这是最常见的。状态字节 0xAn 表示独立应用每个单独键的接触后。

通常,键盘上都有一些用于进一步控制声音的刻度盘或开关。这些装置被称为"控制器",所有变化都由状态字节 0xBn 表示。通过 0 ~ 121 的号码确认控制器。0xBn 状态字节也用于 Channel Mode 消息,这些消息显示了合成器如何在通道中响应同时发生的

音符。

一个非常重要的控制器是上下转换音调的轮，它有一个单独的 MIDI 消息，其状态字节是 0xEn。

表 22-2 中所缺少的是状态字节以 F0 ~ FF 开始的消息。这些消息被称为系统消息，因为它们适用于整个 MIDI 系统，而不是部分通道。系统消息通常用于同步的目的、引发音序器、重新设置硬件，以及获得信息。

表 22-2　MIDI 通道消息(n = 通道号码，从 0 到 15)

MIDI 消息	数据字节	值
Note Off	8n kk vv	kk = 键号(0 ~ 127)
		vv = 速度(0 ~ 127)
Note On	9n kk vv	kk = 键号(0 ~ 127)
		vv = 速度(1 ~ 127, 0 = note off)
Polyphonic After Touch	An kk tt	kk = 键号(0 ~ 127)
		tt = 按下之后 (0 ~ 127)
Control Change	Bn cc xx	cc = 控制器(0 ~ 121)
		xx = 值(0 ~ 127)
Channel Mode Local Control	Bn 7A xx	xx = 0(关)，127(开)
All Notes Off	Bn 7B 00	
Omni Mode Off	Bn 7C 00	
Omni Mode On	Bn 7D 00	
Mono Mode On	Bn 7E cc	cc = 频道数
Poly Mode On	Bn 7F 00	
Program Change	Cn pp	pp = 程序(0 ~ 127)
Channel After Touch	Dn tt	tt = 按下之后(0 ~ 127)
Pitch Wheel Change	En ll hh	ll = 低 7 位(0 ~ 127)
		hh = 高 7 位(0 ~ 127)

许多 MIDI 控制器连续发送状态字节 0xFE，该字节称为 Active Sensing 消息。这简单地表示了 MIDI 控制器仍依附于系统。

一个重要的系统消息是以状态字节 0xF0 开始的“系统专用”消息。此消息用于将数据块按厂商与合成器所依靠的格式传递给合成器。(例如，用这种方法可以将新的声音定义从计算机传递给合成器。)系统专用消息只是可以包含多于 2 个数据字节的唯一消息。实际上，数据字节数是变化的，而每个数据字节的高位都设置为 0。状态字节 0xF7 表示系统专用消息的结尾。

系统专用消息也用于从合成器转储数据(例如，声音定义)。这些数据都是通过

MIDI Out 端口来自合成器。如果要用设备无关的方式对 MIDI 编程,则应该尽可能避免使用系统专用消息,但是它们对于定义新的合成器声音是非常有用的。

MIDI 文件(扩展名是.MDI)是带有定时信息的 MIDI 信息集,可以用 MCI 播放 MIDI 文件。不过,我将在本章的后面讨论低级 midiOut 函数。

22.3.5 MIDI 序列简介

低级 MIDI 的 API 包括前缀为 midiIn 和 midiOut 的函数,它们分别用于读取来自外部控制器的 MIDI 序列和在内部或外部的合成器上播放音乐。尽管其名称为"低级",但使用这些函数时并不需要了解 MIDI 卡上的硬件接口。

要在播放音乐的准备期间打开一个 MIDI 输出设备,可以调用 midiOutOpen 函数:

```
error = midiOutOpen (&hMidiOut, wDeviceID, dwCallBack,
                     dwCallBackData, dwFlags) ;
```

如果调用成功,则函数返回 0,否则返回错误代码。如果参数设置正确,则常见的一种错误就是 MIDI 设备已被其他程序使用。

该函数的第一个参数是指向 HMIDIOUT 类型变量的指针,它接收后面用于 MIDI 输出函数的 MIDI 输出句柄。第二个参数是设备 ID。要使用真实的 MIDI 设备,这个参数范围可以是从 0 到小于由 midiOutGetNumDevs 返回的数。您还可以使用 MIDIMAPPER,它在 MMSYSTEM.H 中定义为 -1。多数情况下,函数的后三个参数设置为 NULL 或 0。

一旦打开一个 MIDI 输出设备并获得其句柄,您就可以向该设备发送 MIDI 消息。这时可以调用:

```
error = midiOutShortMsg (hMidiOut, dwMessage) ;
```

第一个参数是从 midiOutOpen 函数获得的句柄。第二个参数是包装在 32 位 DWORD 中的 1 字节、2 字节或者 3 字节的消息。在前面讨论过,MIDI 消息以状态字节开始,后面是 0、1 或 2 字节的数据。在 dwMessage 中,状态字节是最不重要的,第一个数据字节次之,第二个数据字节再次之,最重要的字节是 0。

例如,要在 MIDI 通道 5 上以 0x7F 的速度演奏中音 C(音符是 0x3C),则需要 3 字节的 Note On 消息:

```
0x95 0x3C 0x7F
```

midiOutShortMsg 的参数 dwMessage 等于 0x007F3C95。

三个基础的 MIDI 消息是 Program Change(可为某一特定通道而改变乐器声音)、Note On 和 Note Off。打开一个 MIDI 输出设备后,应该从一个 Program Change 消息开始,然后发送相同数量的 Note On 和 Note Off 消息。

当您一直演奏要演奏的音乐时,您可以重置 MIDI 输出设备,以确保关闭所有的音

符：

```
midiOutReset (hMidiOut) ;
```

然后关闭设备：

```
midiOutClose (hMidiOut) ;
```

使用低级的 MIDI 输出 API 时，midiOutOpen、midiOutShortMsg、midiOutReset 和 midiOutClose 是您需要的 4 个基础函数。

现在让我们演奏一段音乐。BACHTOCC 程序，如程序 22-8 所示，演奏了 J. S. Bach 著名的风琴演奏 D 小调《Toccata and Fugue》中托卡塔部分的第一小节。

程序 22-8　BACHTOCC 程序

BACHTOCC.C

```
/*-----------------------
    BACHTOCC.C——Bach Toccata in D Minor (First Bar)
                (c) Charles Petzold, 1998
   -----------------------*/

#include <windows.h>

#define ID_TIMER        1

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

TCHAR szAppName[] = TEXT ("BachTocc") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;
```

```
    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                    szAppName, MB_ICONERROR) ;
        return 0 ;
    }

    hwnd = CreateWindow (szAppName,
                         TEXT ("Bach Toccata in D Minor (First Bar)"),
                         WS_OVERLAPPEDWINDOW,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         CW_USEDEFAULT, CW_USEDEFAULT,
                         NULL, NULL, hInstance, NULL) ;
    if (!hwnd)
        return 0 ;

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

DWORD MidiOutMessage (HMIDIOUT hMidi, int iStatus, int iChannel,
                                      int iData1, int iData2)
{
    DWORD dwMessage = iStatus | iChannel | (iData1 << 8) | (iData2 << 16) ;

    return midiOutShortMsg (hMidi, dwMessage) ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static struct
    {
        int iDur ;
        int iNote [2] ;
    }
    noteseq [] = { 110,  69,  81,  110,  67,  79,  990,  69,  81,  220,  -1,  -1,
                   110,  67,  79,  110,  65,  77,  110,  64,  76,  110,  62,  74,
                   220,  61,  73,  440,  62,  74, 1980,  -1,  -1,  110,  57,  69,
                   110,  55,  67,  990,  57,  69,  220,  -1,  -1,  220,  52,  64,
                   220,  53,  65,  220,  49,  61,  440,  50,  62, 1980,  -1,  -1 } ;
```

```
static HMIDIOUT  hMidiOut ;
static int       iIndex ;
int              i ;

switch (message)
{
case WM_CREATE:
        // Open MIDIMAPPER device

    if (midiOutOpen (&hMidiOut, MIDIMAPPER, 0, 0, 0))
    {
        MessageBeep (MB_ICONEXCLAMATION) ;
        MessageBox (hwnd, TEXT ("Cannot open MIDI output device!"),
                          szAppName, MB_ICONEXCLAMATION | MB_OK) ;
        return -1 ;
    }
        // Send Program Change messages for "Church Organ"

    MidiOutMessage (hMidiOut, 0xC0, 0, 19, 0) ;
    MidiOutMessage (hMidiOut, 0xC0, 12, 19, 0) ;

    SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
    return 0 ;

case WM_TIMER:
        // Loop for 2-note polyphony

    for (i = 0 ; i < 2 ; i++)
    {
            // Note Off messages for previous note

        if (iIndex != 0 && noteseq[iIndex - 1].iNote[i] != -1)
        {
            MidiOutMessage (hMidiOut, 0x80, 0,
                            noteseq[iIndex - 1].iNote[i], 0) ;

            MidiOutMessage (hMidiOut, 0x80, 12,
                            noteseq[iIndex - 1].iNote[i], 0) ;
        }
            // Note On messages for new note

        if (iIndex != sizeof (noteseq) / sizeof (noteseq[0]) &&
            noteseq[iIndex].iNote[i] != -1)
        {
            MidiOutMessage (hMidiOut, 0x90, 0,
                            noteseq[iIndex].iNote[i], 127) ;

            MidiOutMessage (hMidiOut, 0x90, 12,
```

```
                    noteseq[iIndex].iNote[i], 127) ;
            }
        }

        if (iIndex != sizeof (noteseq) / sizeof (noteseq[0]))
        {
            SetTimer (hwnd, ID_TIMER, noteseq[iIndex++].iDur - 1, NULL) ;
        }
        else
        {
            KillTimer (hwnd, ID_TIMER) ;
            DestroyWindow (hwnd) ;
        }
        return 0 ;

    case WM_DESTROY:
        midiOutReset (hMidiOut) ;
        midiOutClose (hMidiOut) ;
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

图 22-1 显示了 Bach 的 D 小调 Toccata 的第一小节。

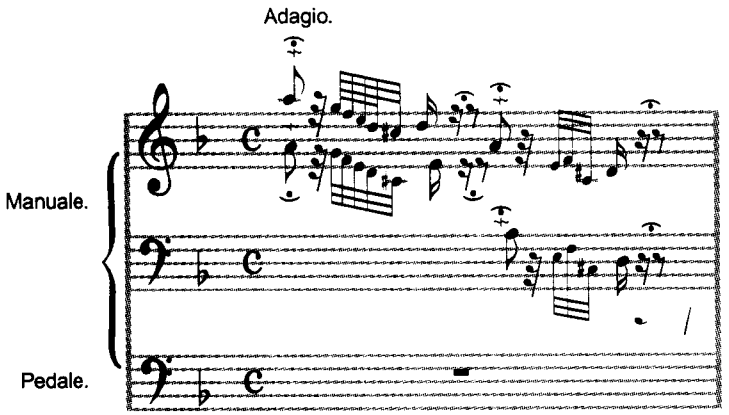


图 22-1　Bach 的 D 小调 Toccata 和 Fugue 的第一小节

在这里要做的就是把音乐转换成一系列的数字——基本键号和定时信息，其中定时信息表示发送 Note On(对应于风琴键按下)和 Note Off(释放键)消息的时间。由于风琴键盘对速度不敏感，所以我们用相同的速度来演奏所有的音符。另外一个简化是忽略断奏(在连续的音符之间留下一个很短的停顿，以达到尖硬的效果)和连奏(在连续的

音符之间有更圆润的重叠)之间的区别。我们假定一个音符结束后面紧接着下一个音符开始。

如果看得懂乐谱,那么您就会注意到托卡塔曲以两个平行的八度音阶开始。因此BACHTOCC创建了一个数据结构noteseq来存储一系列的音符持续时间以及两个键号。遗憾的是,音乐持续进入第二小节就需要更不能适应特殊环境的方法来存储此信息。我将四分音符的持续时间定义为1760毫秒,也就是说,八分音符(在音符或者休止符上有一个符尾)的持续时间是880毫秒,十六分音符(两个符尾)是440毫秒,三十二分音符(三个符尾)是220毫秒,六十四分音符(四个符尾)是110毫秒。

这第一小节中有两个波音——一个在第一个音符处,另一个在小节的中间。这在乐谱上用带一条短竖线的曲线表示。在结构复杂的乐曲中,波音符号表示此音符实际应演奏为三个音符——标出的音符、比它低一个全音的音符,然后还是标出的音符。前两个音符演奏得要快,第三个音符要持续剩余的时间。例如,第一个音符是带波音的A,则应演奏为A、G、A。我将波音的前两个音符定义为六十四分音符,所以每个音符都持续110毫秒。

在第一小节还有四个延长符号。乐谱上表示为中间带点的半圆形。延长符号表示该音符在演奏时所持续的时间比标记的时间要长,通常由演奏者决定具体的时间。我对于延长符号延长了50%的时间。

可以看到,即使转换一小段看来简单直接的乐曲,例如D小调《Toccata》的开头,也并不是件容易的事!

noteseq结构数组包含了这一小节中平行的音符和休止符的三个数字。音符持续时间的后面是用于平行八度音阶的两个MIDI键号。例如,第一个音符是A,持续时间是110毫秒。因为中音C的MIDI键号是60,所以中音C上面的A的键号是69,比A高一个八度音阶的键号是81。因此,noteseq数组的前三个数是110、69和81。我用音符值-1表示休止符。

WM_CREATE消息期间,BACHTOCC设置一个Windows计时器用于计时1000毫秒——表示乐曲从第1秒开始演奏——然后用MIDIMAPPER设备ID调用midiOutOpen。

BACHTOCC只需要一种乐器(风琴)的声音,所以只需要一个通道。为了简化MIDI消息的发送,BACHTOCC中还定义了一个小函数MidiOutMessage。此函数接收MIDI输出句柄、状态字节、通道号,以及两个字节数据,其功能是把这些数字打包到一个32位的消息并调用midiOutShortMsg。

在WM_CREATE消息处理过程的后期,BACHTOCC发送一个Program Change消息来选择"教堂风琴"的声音。在General MIDI声音分配中,教堂风琴声音在Program Change消息中用数字字节19表示。实际演奏的音符出现在WM_TIMER消息期间。用循环来处理两个音符的多音。如果前一个音符还在演奏,BACHTOCC就为该音符发送Note Off消息。然后,如果下一个音符不是休止符,则向通道0和12发送Note On消息。

随后，重置 Windows 计时器，使其与 noteseq 结构中音符的持续时间一致。

音乐演奏完后，BACHTOCC 删除窗口。在 WM_DESTROY 消息期间，程序调用 midiOutReset 和 midiOutClose，然后终止程序。

尽管 BACHTOCC 合理地处理和计算声音（如果还不完全像真人演奏风琴），但一般情况下用 Windows 计时器按这种方式来演奏音乐并不管用。问题在于 Windows 计时器是基于 PC 的系统时钟，其分辨率不能满足音乐的要求。而且，Windows 计时器不是异步的。这样，如果其他程序正忙于运行，则获得 WM_TIMER 消息就会有轻微的延迟。如果程序不能立即处理这些消息，则就会放弃 WM_TIMER 消息，这时的声音听起来一团糟。

因此，当 BACHTOCC 显示了如何调用低级 MIDI 输出函数时，使用 Windows 计时器显然不适合精确的音乐创作。所以，Windows 还提供了一系列附加的计时器函数，使用低级的 MIDI 输出函数时可以利用这些函数。这些函数的前缀为 time，您可以利用它们将计时器的分辨率设置到最低 1 毫秒。我将在本章结尾的 DRUM 程序向您演示使用这些函数的方法。

22.3.6　通过键盘演奏 MIDI 合成器

因为大多数 PC 用户可能都没有连接在机器上的 MIDI 键盘，所以可以用每个人都有的键盘（上面全部的字母键和数据键）来代替。程序 22-9 所示的 KBMIDI 程序允许您用 PC 键盘来演奏电子音乐合成器——不管是连接在声卡上的，还是挂接在 MIDI Out 端口的外部合成器。KBMIDI 让您完全控制 MIDI 输出设备（内部或外部的合成器）、MIDI 通道和乐器声音。除了演奏时的趣味性以外，我还发现此程序对于探究 Windows 如何实现 MIDI 支持很有用。

程序 22-9　KBMIDI 程序

KBMIDI.C

```
/*------------------------
   KBMIDI.C——Keyboard MIDI Player
            (c) Charles Petzold, 1998
   ------------------------*/

#include <windows.h>

// Defines for Menu IDs
// --------------------

#define IDM_OPEN      0x100
#define IDM_CLOSE     0x101
```

```
#define IDM_DEVICE      0x200
#define IDM_CHANNEL     0x300
#define IDM_VOICE       0x400

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM);

TCHAR       szAppName [] = TEXT ("KBMidi") ;
HMIDIOUT    hMidiOut ;
int         iDevice = MIDIMAPPER, iChannel = 0, iVoice = 0, iVelocity = 64 ;
int         cxCaps, cyChar, xOffset, yOffset ;

    // Structures and data for showing families and instruments on menu
    // - - - - - - - - - - - - - - - - - - - - - - - - - - - -

typedef struct
{
     TCHAR * szInst ;
     int     iVoice ;
}
INSTRUMENT ;

typedef struct
{
     TCHAR       * szFam ;
     INSTRUMENT    inst [8] ;
}
FAMILY ;

FAMILY fam [16] = {

     TEXT ("Piano"),

          TEXT ("Acoustic Grand Piano"),          0,
          TEXT ("Bright Acoustic Piano"),         1,
          TEXT ("Electric Grand Piano"),          2,
          TEXT ("Honky-tonk Piano"),              3,
          TEXT ("Rhodes Piano"),                  4,
          TEXT ("Chorused Piano"),                5,
          TEXT ("Harpsichord"),                   6,
          TEXT ("Clavinet"),                      7,

     TEXT ("Chromatic Percussion"),

          TEXT ("Celesta"),                       8,
          TEXT ("Glockenspiel"),                  9,
          TEXT ("Music Box"),                    10,
          TEXT ("Vibraphone"),                   11,
          TEXT ("Marimba"),                      12,
```

```
        TEXT ("Xylophone"),                 13,
        TEXT ("Tubular Bells"),             14,
        TEXT ("Dulcimer"),                  15,

    TEXT ("Organ"),

        TEXT ("Hammond Organ"),             16,
        TEXT ("Percussive Organ"),          17,
        TEXT ("Rock Organ"),                18,
        TEXT ("Church Organ"),              19,
        TEXT ("Reed Organ"),                20,
        TEXT ("Accordian"),                 21,
        TEXT ("Harmonica"),                 22,
        TEXT ("Tango Accordian"),           23,

    TEXT ("Guitar"),

        TEXT ("Acoustic Guitar (nylon)"),   24,
        TEXT ("Acoustic Guitar (steel)"),   25,
        TEXT ("Electric Guitar (jazz)"),    26,
        TEXT ("Electric Guitar (clean)"),   27,
        TEXT ("Electric Guitar (muted)"),   28,
        TEXT ("Overdriven Guitar"),         29,
        TEXT ("Distortion Guitar"),         30,
        TEXT ("Guitar Harmonics"),          31,

    TEXT ("Bass"),

        TEXT ("Acoustic Bass"),             32,
        TEXT ("Electric Bass (finger)"),    33,
        TEXT ("Electric Bass (pick)"),      34,
        TEXT ("Fretless Bass"),             35,
        TEXT ("Slap Bass 1"),               36,
        TEXT ("Slap Bass 2"),               37,
        TEXT ("Synth Bass 1"),              38,
        TEXT ("Synth Bass 2"),              39,

    TEXT ("Strings"),

        TEXT ("Violin"),                    40,
        TEXT ("Viola"),                     41,
        TEXT ("Cello"),                     42,
        TEXT ("Contrabass"),                43,
        TEXT ("Tremolo Strings"),           44,
        TEXT ("Pizzicato Strings"),         45,
        TEXT ("Orchestral Harp"),           46,
        TEXT ("Timpani"),                   47,

    TEXT ("Ensemble"),
```

```
        TEXT ("String Ensemble 1"),          48,
        TEXT ("String Ensemble 2"),          49,
        TEXT ("Synth Strings 1"),            50,
        TEXT ("Synth Strings 2"),            51,
        TEXT ("Choir Aahs"),                 52,
        TEXT ("Voice Oohs"),                 53,
        TEXT ("Synth Voice"),                54,
        TEXT ("Orchestra Hit"),              55,

    TEXT ("Brass"),

        TEXT ("Trumpet"),                    56,
        TEXT ("Trombone"),                   57,
        TEXT ("Tuba"),                       58,
        TEXT ("Muted Trumpet"),              59,
        TEXT ("French Horn"),                60,
        TEXT ("Brass Section"),              61,
        TEXT ("Synth Brass 1"),              62,
        TEXT ("Synth Brass 2"),              63,

    TEXT ("Reed"),

        TEXT ("Soprano Sax"),                64,
        TEXT ("Alto Sax"),                   65,
        TEXT ("Tenor Sax"),                  66,
        TEXT ("Baritone Sax"),               67,
        TEXT ("Oboe"),                       68,
        TEXT ("English Horn"),               69,
        TEXT ("Bassoon"),                    70,
        TEXT ("Clarinet"),                   71,

    TEXT ("Pipe"),

        TEXT ("Piccolo"),                    72,
        TEXT ("Flute "),                     73,
        TEXT ("Recorder"),                   74,
        TEXT ("Pan Flute"),                  75,
        TEXT ("Bottle Blow"),                76,
        TEXT ("Shakuhachi"),                 77,
        TEXT ("Whistle"),                    78,
        TEXT ("Ocarina"),                    79,

    TEXT ("Synth Lead"),

        TEXT ("Lead 1 (square)"),            80,
        TEXT ("Lead 2 (sawtooth)"),          81,
        TEXT ("Lead 3 (caliope lead)"),      82,
        TEXT ("Lead 4 (chiff lead)"),        83,
```

```
        TEXT ("Lead 5 (charang)"),          84,
        TEXT ("Lead 6 (voice)"),            85,
        TEXT ("Lead 7 (fifths)"),           86,
        TEXT ("Lead 8 (brass + lead)"),     87,

    TEXT ("Synth Pad"),

        TEXT ("Pad 1 (new age)"),           88,
        TEXT ("Pad 2 (warm)"),              89,
        TEXT ("Pad 3 (polysynth)"),         90,
        TEXT ("Pad 4 (choir)"),             91,
        TEXT ("Pad 5 (bowed)"),             92,
        TEXT ("Pad 6 (metallic)"),          93,
        TEXT ("Pad 7 (halo)"),              94,
        TEXT ("Pad 8 (sweep)"),             95,

    TEXT ("Synth Effects"),

        TEXT ("FX 1 (rain)"),               96,
        TEXT ("FX 2 (soundtrack)"),         97,
        TEXT ("FX 3 (crystal)"),            98,
        TEXT ("FX 4 (atmosphere)"),         99,
        TEXT ("FX 5 (brightness)"),         100,
        TEXT ("FX 6 (goblins)"),            101,
        TEXT ("FX 7 (echoes)"),             102,
        TEXT ("FX 8 (sci-fi)"),             103,

    TEXT ("Ethnic"),

        TEXT ("Sitar"),                     104,
        TEXT ("Banjo"),                     105,
        TEXT ("Shamisen"),                  106,
        TEXT ("Koto"),                      107,
        TEXT ("Kalimba"),                   108,
        TEXT ("Bagpipe"),                   109,
        TEXT ("Fiddle"),                    110,
        TEXT ("Shanai"),                    111,

    TEXT ("Percussive"),

        TEXT ("Tinkle Bell"),               112,
        TEXT ("Agogo"),                     113,
        TEXT ("Steel Drums"),               114,
        TEXT ("Woodblock"),                 115,
        TEXT ("Taiko Drum"),                116,
        TEXT ("Melodic Tom"),               117,
        TEXT ("Synth Drum"),                118,
        TEXT ("Reverse Cymbal"),            119,
```

```
    TEXT ("Sound Effects"),

        TEXT ("Guitar Fret Noise"),     120,
        TEXT ("Breath Noise"),          121,
        TEXT ("Seashore"),              122,
        TEXT ("Bird Tweet"),            123,
        TEXT ("Telephone Ring"),        124,
        TEXT ("Helicopter"),            125,
        TEXT ("Applause"),              126,
        TEXT ("Gunshot"),               127 } ;

    // Data for translating scan codes to octaves and notes
    // - - - - - - - - - - - - - - - - - - - -

# define NUMSCANS     (sizeof key / sizeof key[0])
struct
{
    int         iOctave ;
    int         iNote ;
    int         yPos ;
    int         xPos ;
    TCHAR * szKey ;
}
key [] =
{
                                                    //  Scan    Char    Oct     Note
                                                    //  - - - - - - - - - - - - - - -
        -1,   -1,   -1,   -1,   NULL,               //  0       None
        -1,   -1,   -1,   -1,   NULL,               //  1       Esc
        -1,   -1,    0,    0,   TEXT (" "),        //  2       1
         5,    1,    0,    2,   TEXT ("C#"),       //  3       2       5       C#
         5,    3,    0,    4,   TEXT ("D#"),       //  4       3       5       D#
        -1,   -1,    0,    6,   TEXT (" "),        //  5       4
         5,    6,    0,    8,   TEXT ("F#"),       //  6       5       5       F#
         5,    8,    0,   10,   TEXT ("G#"),       //  7       6       5       G#
         5,   10,    0,   12,   TEXT ("A#"),       //  8       7       5       A#
        -1,   -1,    0,   14,   TEXT (" "),        //  9       8
         6,    1,    0,   16,   TEXT ("C#"),       //  10      9       6       C#
         6,    3,    0,   18,   TEXT ("D#"),       //  11      0       6       D#
        -1,   -1,    0,   20,   TEXT (" "),        //  12      -
         6,    6,    0,   22,   TEXT ("F#"),       //  13      =       6       F#
        -1,   -1,   -1,   -1,   NULL,               //  14      Back
```

```
    -1,   -1,   -1,   -1,   NULL,              //  15   Tab
     5,    0,    1,    1,   TEXT ("C"),        //  16   q      5      C
     5,    2,    1,    3,   TEXT ("D"),        //  17   w      5      D
     5,    4,    1,    5,   TEXT ("E"),        //  18   e      5      E
     5,    5,    1,    7,   TEXT ("F"),        //  19   r      5      F
     5,    7,    1,    9,   TEXT ("G"),        //  20   t      5      G
     5,    9,    1,   11,   TEXT ("A"),        //  21   y      5      A
     5,   11,    1,   13,   TEXT ("B"),        //  22   u      5      B
     6,    0,    1,   15,   TEXT ("C"),        //  23   i      6      C
     6,    2,    1,   17,   TEXT ("D"),        //  24   o      6      D
     6,    4,    1,   19,   TEXT ("E"),        //  25   p      6      E
     6,    5,    1,   21,   TEXT ("F"),        //  26   [      6      F
     6,    7,    1,   23,   TEXT ("G"),        //  27   ]      6      G
    -1,   -1,   -1,   -1,   NULL,              //  28   Ent

    -1,   -1,   -1,   -1,   NULL,              //  29   Ctrl
     3,    8,    2,    2,   TEXT ("G#"),       //  30   a      3      G#
     3,   10,    2,    4,   TEXT ("A#"),       //  31   s      3      A#
    -1,   -1,    2,    6,   TEXT (""),         //  32   d
     4,    1,    2,    8,   TEXT ("C#"),       //  33   f      4      C#
     4,    3,    2,   10,   TEXT ("D#"),       //  34   g      4      D#
    -1,   -1,    2,   12,   TEXT (""),         //  35   h
     4,    6,    2,   14,   TEXT ("F#"),       //  36   j      4      F#
     4,    8,    2,   16,   TEXT ("G#"),       //  37   k      4      G#
     4,   10,    2,   18,   TEXT ("A#"),       //  38   l      4      A#
    -1,   -1,    2,   20,   TEXT (""),         //  39   ;
     5,    1,    2,   22,   TEXT ("C#"),       //  40   '      5      C#
    -1,   -1,   -1,   -1,   NULL,              //  41   `

    -1,   -1,   -1,   -1,   NULL,              //  42   Shift
    -1,   -1,   -1,   -1,   NULL,              //  43   \      (not line continuation)

     3,    9,    3,    3,   TEXT ("A"),        //  44   z      3      A
     3,   11,    3,    5,   TEXT ("B"),        //  45   x      3      B
     4,    0,    3,    7,   TEXT ("C"),        //  46   c      4      C
     4,    2,    3,    9,   TEXT ("D"),        //  47   v      4      D
     4,    4,    3,   11,   TEXT ("E"),        //  48   b      4      E
     4,    5,    3,   13,   TEXT ("F"),        //  49   n      4      F
     4,    7,    3,   15,   TEXT ("G"),        //  50   m      4      G
     4,    9,    3,   17,   TEXT ("A"),        //  51   ,      4      A
     4,   11,    3,   19,   TEXT ("B"),        //  52   .      4      B
     5,    0,    3,   21,   TEXT ("C")         //  53   /      5      C
};
```

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     MSG         msg;
     HWND        hwnd ;
     WNDCLASS    wndclass ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Keyboard MIDI Player"),
                          WS_OVERLAPPEDWINDOW | WS_HSCROLL | WS_VSCROLL,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;
     if (!hwnd)
          return 0 ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd);

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

// Create the program's menu (called from WndProc, WM_CREATE)
// - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
HMENU CreateTheMenu (int iNumDevs)
{
     TCHAR           szBuffer [32] ;
     HMENU           hMenu, hMenuPopup, hMenuSubPopup ;
     int             i, iFam, iIns ;
     MIDIOUTCAPS     moc ;

     hMenu = CreateMenu () ;

          // Create "On/Off" popup menu

     hMenuPopup = CreateMenu () ;

     AppendMenu (hMenuPopup, MF_STRING, IDM_OPEN, TEXT ("&Open")) ;
     AppendMenu (hMenuPopup, MF_STRING | MF_CHECKED, IDM_CLOSE,
                             TEXT ("&Closed")) ;

     AppendMenu (hMenu, MF_STRING | MF_POPUP, (UINT) hMenuPopup,
                        TEXT ("&Status")) ;

          // Create "Device" popup menu

     hMenuPopup = CreateMenu () ;

          // Put MIDI Mapper on menu if it's installed

     if (!midiOutGetDevCaps (MIDIMAPPER, &moc, sizeof (moc)))
          AppendMenu (hMenuPopup, MF_STRING, IDM_DEVICE + (int) MIDIMAPPER,
                      moc.szPname) ;
     else
          iDevice = 0 ;

          // Add the rest of the MIDI devices

     for (i = 0 ; i < iNumDevs ; i++)
     {
          midiOutGetDevCaps (i, &moc, sizeof (moc)) ;
          AppendMenu (hMenuPopup, MF_STRING, IDM_DEVICE + i, moc.szPname) ;
     }

     CheckMenuItem (hMenuPopup, 0, MF_BYPOSITION | MF_CHECKED) ;
     AppendMenu (hMenu, MF_STRING | MF_POPUP, (UINT) hMenuPopup,
                        TEXT ("&Device")) ;

          // Create "Channel" popup menu

     hMenuPopup = CreateMenu () ;
```

```
    for (i = 0 ; i < 16 ; i++)
    {
         wsprintf (szBuffer, TEXT ("%d"), i + 1) ;
         AppendMenu (hMenuPopup, MF_STRING | (i ? MF_UNCHECKED : MF_CHECKED),
                                 IDM_CHANNEL + i, szBuffer) ;
    }

    AppendMenu (hMenu, MF_STRING | MF_POPUP, (UINT) hMenuPopup,
                       TEXT ("&Channel")) ;

        // Create "Voice" popup menu

    hMenuPopup = CreateMenu () ;

    for (iFam = 0 ; iFam < 16 ; iFam++)
    {
         hMenuSubPopup = CreateMenu () ;

         for (iIns = 0 ; iIns < 8 ; iIns++)
         {
              wsprintf (szBuffer, TEXT ("&%d. \t%s"), iIns + 1,
                                  fam[iFam].inst[iIns].szInst) ;
              AppendMenu (hMenuSubPopup,
                          MF_STRING | (fam[iFam].inst[iIns].iVoice ?
                                              MF_UNCHECKED : MF_CHECKED),
                          fam[iFam].inst[iIns].iVoice + IDM_VOICE,
                          szBuffer) ;
         }

         wsprintf (szBuffer, TEXT ("&%c. \t%s"), 'A' + iFam,
                             fam[iFam].szFam) ;
         AppendMenu (hMenuPopup, MF_STRING | MF_POPUP, (UINT) hMenuSubPopup,
                                 szBuffer) ;
    }
    AppendMenu (hMenu, MF_STRING | MF_POPUP, (UINT) hMenuPopup,
                       TEXT ("&Voice")) ;
    return hMenu ;
}

// Routines for simplifying MIDI output
// - - - - - - - - - - - - - - -

DWORD MidiOutMessage (HMIDIOUT hMidi, int iStatus, int iChannel,
                      int iData1, int iData2)
{
     DWORD dwMessage ;
```

```
    dwMessage = iStatus | iChannel | (iData1 << 8) | (iData2 << 16) ;

    return midiOutShortMsg (hMidi, dwMessage) ;
}

DWORD MidiNoteOff (HMIDIOUT hMidi, int iChannel, int iOct, int iNote, int iVel)
{
    return MidiOutMessage (hMidi, 0x080, iChannel, 12 * iOct + iNote, iVel) ;
}

DWORD MidiNoteOn (HMIDIOUT hMidi, int iChannel, int iOct, int iNote, int iVel)
{
    return MidiOutMessage (hMidi, 0x090, iChannel, 12 * iOct + iNote, iVel) ;
}

DWORD MidiSetPatch (HMIDIOUT hMidi, int iChannel, int iVoice)
{
    return MidiOutMessage (hMidi, 0x0C0, iChannel, iVoice, 0) ;
}
DWORD MidiPitchBend (HMIDIOUT hMidi, int iChannel, int iBend)
{
    return MidiOutMessage (hMidi, 0x0E0, iChannel, iBend & 0x7F, iBend >> 7) ;
}

// Draw a single key on window
// - - - - - - - - - - - - -

VOID DrawKey (HDC hdc, int iScanCode, BOOL fInvert)
{
    RECT rc ;
    rc.left   = 3 * cxCaps * key[iScanCode].xPos / 2 + xOffset ;
    rc.top    = 3 * cyChar * key[iScanCode].yPos / 2 + yOffset ;
    rc.right  = rc.left + 3 * cxCaps ;
    rc.bottom = rc.top + 3 * cyChar / 2 ;

    SetTextColor (hdc, fInvert ? 0x00FFFFFFul : 0x00000000ul) ;
    SetBkColor   (hdc, fInvert ? 0x00000000ul : 0x00FFFFFFul) ;

    FillRect (hdc, &rc, GetStockObject (fInvert ? BLACK_BRUSH : WHITE_BRUSH)) ;

    DrawText (hdc, key[iScanCode].szKey, -1, &rc,
              DT_SINGLELINE | DT_CENTER | DT_VCENTER) ;

    FrameRect (hdc, &rc, GetStockObject (BLACK_BRUSH)) ;
}

// Process a Key Up or Key Down message
// - - - - - - - - - - - - - - - - -
```

```
VOID ProcessKey (HDC hdc, UINT message, LPARAM lParam)
{
     int iScanCode, iOctave, iNote ;

     iScanCode = 0x0FF & HIWORD (lParam) ;

     if (iScanCode >= NUMSCANS)                              // No scan codes over 53
          return ;

     if ((iOctave = key[iScanCode].iOctave) == -1)           // Non-music key
          return ;

     if (GetKeyState (VK_SHIFT) < 0)
          iOctave += 0x20000000 & lParam ? 2 : 1 ;

     if (GetKeyState (VK_CONTROL) < 0)
          iOctave -= 0x20000000 & lParam ? 2 : 1 ;

     iNote = key[iScanCode].iNote ;

     if (message == WM_KEYUP)                                // For key up
     {
          MidiNoteOff (hMidiOut, iChannel, iOctave, iNote, 0) ;   // Note off
          DrawKey (hdc, iScanCode, FALSE) ;
          return ;
     }

     if (0 x40000000 & lParam)                               // ignore typematics
          return ;

     MidiNoteOn (hMidiOut, iChannel, iOctave, iNote, iVelocity) ;   // Note on
     DrawKey (hdc, iScanCode, TRUE) ;                        // Draw the inverted key
}

// Window Procedure
// - - - - - - - -

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BOOL     bOpened = FALSE ;
     HDC             hdc ;
     HMENU           hMenu ;
     int             i, iNumDevs, iPitchBend, cxClient, cyClient ;
     MIDIOUTCAPS     moc ;
     PAINTSTRUCT     ps ;
     SIZE            size ;
     TCHAR           szBuffer [16] ;
```

```
switch (message)
{
case WM_CREATE:
        // Get size of capital letters in system font

    hdc = GetDC (hwnd) ;

    GetTextExtentPoint (hdc, TEXT ("M"), 1, &size) ;
    cxCaps = size.cx ;
    cyChar = size.cy ;

    ReleaseDC (hwnd, hdc) ;

        // Initialize "Volume" scroll bar

    SetScrollRange (hwnd, SB_HORZ, 1, 127, FALSE) ;
    SetScrollPos (hwnd, SB_HORZ, iVelocity, TRUE) ;

        // Initialize "Pitch Bend" scroll bar

    SetScrollRange (hwnd, SB_VERT, 0, 16383, FALSE) ;
    SetScrollPos (hwnd, SB_VERT, 8192, TRUE) ;

        // Get number of MIDI output devices and set up menu

    if (0 == (iNumDevs = midiOutGetNumDevs ()))
    {
        MessageBeep (MB_ICONSTOP) ;
        MessageBox (hwnd, TEXT ("No MIDI output devices!"),
                    szAppName, MB_OK | MB_ICONSTOP) ;
        return -1 ;
    }

    SetMenu (hwnd, CreateTheMenu (iNumDevs)) ;
    return 0 ;

case WM_SIZE:
    cxClient = LOWORD (lParam) ;
    cyClient = HIWORD (lParam) ;

    xOffset = (cxClient - 25 * 3 * cxCaps / 2) / 2 ;
    yOffset = (cyClient - 11 * cyChar) / 2 + 5 * cyChar ;
    return 0 ;

case WM_COMMAND:
    hMenu = GetMenu (hwnd) ;

        // "Open" menu command
```

```
if (LOWORD (wParam) == IDM_OPEN && !bOpened)
{
     if (midiOutOpen (&hMidiOut, iDevice, 0, 0, 0))
     {
          MessageBeep (MB_ICONEXCLAMATION) ;
          MessageBox (hwnd, TEXT ("Cannot open MIDI device"),
                      szAppName, MB_OK | MB_ICONEXCLAMATION) ;
     }
     else
     {
          CheckMenuItem (hMenu, IDM_OPEN, MF_CHECKED) ;
          CheckMenuItem (hMenu, IDM_CLOSE, MF_UNCHECKED) ;

          MidiSetPatch (hMidiOut, iChannel, iVoice) ;
          bOpened = TRUE ;
     }
}

    // "Close" menu command

else if (LOWORD (wParam) == IDM_CLOSE && bOpened)
{
     CheckMenuItem (hMenu, IDM_OPEN, MF_UNCHECKED) ;
     CheckMenuItem (hMenu, IDM_CLOSE, MF_CHECKED) ;

          // Turn all keys off and close device

     for (i = 0 ; i < 16 ; i++)
          MidiOutMessage (hMidiOut, 0xB0, i, 123, 0) ;

     midiOutClose (hMidiOut) ;
     bOpened = FALSE ;
}

    // Change MIDI "Device" menu command

else if (LOWORD (wParam) >= IDM_DEVICE - 1 &&
         LOWORD (wParam) < IDM_CHANNEL)
{
     CheckMenuItem (hMenu, IDM_DEVICE + iDevice, MF_UNCHECKED) ;
     iDevice = LOWORD (wParam) - IDM_DEVICE ;
     CheckMenuItem (hMenu, IDM_DEVICE + iDevice, MF_CHECKED) ;

          // Close and reopen MIDI device

     if (bOpened)
     {
```

```
                SendMessage (hwnd, WM_COMMAND, IDM_CLOSE, 0L) ;
                SendMessage (hwnd, WM_COMMAND, IDM_OPEN, 0L) ;
            }
        }

        // Change MIDI "Channel" menu command

        else if (LOWORD (wParam) >= IDM_CHANNEL &&
                 LOWORD (wParam) < IDM_VOICE)
        {
            CheckMenuItem (hMenu, IDM_CHANNEL + iChannel, MF_UNCHECKED);
            iChannel = LOWORD (wParam) - IDM_CHANNEL ;
            CheckMenuItem (hMenu, IDM_CHANNEL + iChannel, MF_CHECKED) ;

            if (bOpened)
                MidiSetPatch (hMidiOut, iChannel, iVoice) ;
        }

        // Change MIDI "Voice" menu command

        else if (LOWORD (wParam) >= IDM_VOICE)
        {
            CheckMenuItem (hMenu, IDM_VOICE + iVoice, MF_UNCHECKED) ;
            iVoice = LOWORD (wParam) - IDM_VOICE ;
            CheckMenuItem (hMenu, IDM_VOICE + iVoice, MF_CHECKED) ;

            if (bOpened)
                MidiSetPatch (hMidiOut, iChannel, iVoice) ;
        }

        InvalidateRect (hwnd, NULL, TRUE) ;
        return 0 ;

        // Process a Key Up or Key Down message

    case WM_KEYUP:
    case WM_KEYDOWN:
        hdc = GetDC (hwnd) ;

        if (bOpened)
            ProcessKey (hdc, message, lParam) ;

        ReleaseDC (hwnd, hdc) ;
        return 0 ;

        // For Escape, turn off all notes and repaint
```

```
case WM_CHAR:
    if (bOpened && wParam == 27)
    {
        for (i = 0 ; i < 16 ; i++)
            MidiOutMessage (hMidiOut, 0xB0, i, 123, 0) ;

        InvalidateRect (hwnd, NULL, TRUE) ;
    }
    return 0 ;

    // Horizontal scroll: Velocity

case WM_HSCROLL:
    switch (LOWORD (wParam))
    {
    case SB_LINEUP:          iVelocity -= 1 ;    break ;
    case SB_LINEDOWN:        iVelocity += 1 ;    break ;
    case SB_PAGEUP:          iVelocity -= 8 ;    break ;
    case SB_PAGEDOWN:        iVelocity += 8 ;    break ;
    case SB_THUMBPOSITION:   iVelocity = HIWORD (wParam) ; break ;
    default:                 return 0 ;
    }
    iVelocity = max (1, min (iVelocity, 127)) ;
    SetScrollPos (hwnd, SB_HORZ, iVelocity, TRUE) ;
    return 0 ;

    // Vertical scroll: Pitch Bend

case WM_VSCROLL:
    switch (LOWORD (wParam))
    {
    case SB_THUMBTRACK:    iPitchBend = 16383 - HIWORD (wParam) ;  break ;
    case SB_THUMBPOSITION: iPitchBend = 8191 ;                     break ;
    default:               return 0 ;
    }
    iPitchBend = max (0, min (iPitchBend, 16383)) ;
    SetScrollPos (hwnd, SB_VERT, 16383 - iPitchBend, TRUE) ;

    if (bOpened)
        MidiPitchBend (hMidiOut, iChannel, iPitchBend) ;
    return 0 ;

case WM_PAINT:
    hdc = BeginPaint (hwnd, &ps) ;

    for (i = 0 ; i < NUMSCANS ; i++)
        if (key[i].xPos != -1)
```

```
                DrawKey (hdc, i, FALSE) ;

            midiOutGetDevCaps (iDevice, &moc, sizeof (MIDIOUTCAPS)) ;
            wsprintf (szBuffer, TEXT ("Channel %i"), iChannel + 1) ;

            TextOut (hdc, cxCaps, 1 * cyChar,
                     bOpened ? TEXT ("Open") : TEXT ("Closed"),
                     bOpened ? 4 : 6) ;
            TextOut (hdc, cxCaps, 2 * cyChar, moc.szPname,
                     lstrlen (moc.szPname)) ;
            TextOut (hdc, cxCaps, 3 * cyChar, szBuffer, lstrlen (szBuffer)) ;
            TextOut (hdc, cxCaps, 4 * cyChar,
                     fam[iVoice / 8].inst[iVoice % 8].szInst,
                lstrlen (fam[iVoice / 8].inst[iVoice % 8].szInst)) ;

            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY :
            SendMessage (hwnd, WM_COMMAND, IDM_CLOSE, 0L) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}
```

运行 KBMIDI 时，窗口显示了键盘上的键与传统钢琴或风琴按键的对应方式。左下角的 Z 键以 110 Hz 的频率演奏 A。键盘的最下行，右边是中音 C，倒数第二行为其升音或降音。上面两行键继续按此规律变化，从中音 C 到 G#。这样，整个范围是 3 个八度音阶。另外，分别按 Shift 键和 Ctrl 键可使整个音域上升或下降 1 个八度音阶，这样有效的音域就是 5 个八度音阶。

不过，如果立即开始演奏，那么您将听不到任何声音。您必须先从 Status 菜单中选择 Open，打开一个 MIDI 输出设备。如果端口打开成功，则按下一个键，向合成器发送一个 MIDI Note On 消息，释放键则产生一个 Note Off 消息。根据键盘的击键特性，您可以同时演奏几个音符。

从 Status 菜单里选择 Close 来关闭 MIDI 设备。这对于需要在不终止 KBMIDI 程序的情况下运行 Windows 下的其他 MIDI 软件来说是很方便的。

Device 菜单列出了已安装的 MIDI 输出设备，这些设备通过调用 midiOutGetDevCaps 函数获得，其中有些设备可能是 MIDI Out 端口连接的实际存在或不存在的外部合成器。列表还包括 MIDI Mapper 设备。这是从“控制面板”的“多媒体”中选择的 MIDI 合成器。

Channel 菜单用来选择 1 ~ 16 的 MIDI 通道，默认时选择通道 1。KBMIDI 程序产生

的所有 MIDI 消息都发送到所选的通道。

KBMIDI 最后一个菜单项是 Voice，它是一个双层菜单，用于选择 128 种乐器声音，这些声音在 General MIDI 规范中定义，并在 Windows 中实现。这 128 种乐器声音分为 16 个乐器组，每个乐器组有 8 种乐器。由于不同的 MIDI 键号对应于不同的泛音，所以这 128 种乐器声音也称为有旋律的声音。

General MIDI 中还定义了大量无旋律的打击乐器。要演奏打击乐器，可以从 Channel 菜单选择通道 10，还可以从 Voice 菜单选择第一种乐器声音(Acoustic Grand Piano)。这样，按不同的键就可以得到不同打击乐器的声音。从 MIDI 键号 35(低于中音 C 两个八度音阶的 B)到 81(高于中音 C 近两个八度音阶的 A)，共有 47 种不同的打击乐器声音。在下面的 DRUM 程序中就利用了打击乐器通道。

KBMIDI 程序有水平和垂直滚动条。由于 PC 键盘对按键速度不敏感，所以用水平滚动条来控制音符速度。一般来说，这与演奏音符的音量一致。设置完水平滚动条以后，所有的 Note On 消息都将使用这个速度。

垂直滚动条将产生一个被称为“Pitch Bend”的 MIDI 消息。要使用此特性，请按下一个或多个键，然后用鼠标拖动滚动条。向上拖动滚动条音符频率将上升，向下拖动则频率下降，释放滚动条后将恢复正常的基音。

这两个滚动条要小心使用：因为拖动滚动条时，键盘消息将不进入程序的消息循环，因此，如果按下一个键后就开始拖动滚动条，然后在完成拖动之前就释放了该键，那么音符仍将发声。所以，拖动滚动条时不要按下或者释放任何键。对菜单也有类似的规则：按着键时不要进行菜单选择。另外，在按下与释放某个键期间，不要用 Ctrl 键或 Shift 键来改变八度音阶。

如果一个或多个音符出现“粘滞现象”，即释放后继续发声，那么请按下 Esc 键。按下此键将通过向 MIDI 合成器的 16 个通道发送 16 个 All Notes Off 消息来关闭声音。

KBMIDI 没有资源描述文件，而是自己创建菜单。设备名称从 midiOutGetDevCaps 函数获得，乐器种类和名称则存储在程序的一个大数据结构中。

KBMIDI 定义了几个小函数来简化 MIDI 消息。除了 Pitch Bend 消息以外，其他消息在前面都讨论过了。Pitch Bend 消息用两个 7 位值组成一个 14 位的音调弯曲级别：0 ~ 0x1FFF 之间的值降低基音，0x2001 ~ 0x3FFF 之间的值升高基音。

从 Status 菜单选择 Open 时，KBMIDI 为选择的设备调用 midiOutOpen，如调用成功，则调用 MidiSetPatch 函数。设备改变时，KBMIDI 必须关闭前一个设备，必要时再打开新设备。当改变 MIDI 设备、MIDI 通道、乐器声音时，KBMIDI 也必须调用 MidiSetPatch。

KBMIDI 通过处理 WM_KEYUP 消息和 WM_KEYDOWN 消息来控制音符的发音。KBMIDI 中用一个数组把键盘扫描码映射成八度音阶和音符。例如，美国英语键盘上 Z 键的扫描码是 44，数组将其标记为八度音阶是 3，音符是 9(即 A)。在 KBMIDI 的 MidiNoteOn 函数里，这些组合成了 MIDI 键号 45(即 12 乘以 3 再加上 9)。此数据结构也用于

在窗口中画出键——每个键都有特定的水平和垂直位置，以及显示在矩形中的文本字符串。

水平滚动条的处理很直接：所有需要做的就是保存新的速度级，并设置新的滚动条的位置。但是处理垂直滚动条以控制音调弯曲的操作稍有一点特殊，它处理的滚动条命令只有两个：用鼠标拖动滚动条时发生的 SB_THUMBTRACK，以及释放滚动条时的 SB_THUMBPOSITION。处理 SB_THUMBPOSITION 命令时，KBMIDI 将滚动条位置设置为中间级别，并调用 MidiPitchBend，其中参数值是 8192。

22.3.7 MIDI 击鼓器

有些打击乐器，如木琴或定音鼓，是“有旋律的”或“半音阶的”，因为它们可以用不同的音阶演奏乐曲。木琴用木板来对应不同的音阶，定音鼓也可以演奏曲调。这两种乐器及其他的有旋律的打击乐器都可以在 KBMIDI 的 Voice 菜单里选择。

但是，其他许多打击乐器都没有旋律，它们不能调音，而且通常含有太多的噪音，以致不能与某个基音相联系。在“General MIDI”规范中，这些没旋律的打击乐器声在通道 10 有效。不同的键号对应 47 种不同的打击乐器。

DRUM 程序，如程序 22-10 所示，是一个计算机击鼓器。此程序让您用 47 种不同的打击乐器的声音来构造最大到 32 个音符的一个序列，然后在选择的速度和音量下反复演奏这个序列。

程序 22-10　DRUM 程序

DRUM.C

```
/*------------------
   DRUM.C——MIDI Drum Machine
            (c) Charles Petzold, 1998
   ------------------*/

#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>
#include "drumtime.h"
#include "drumfile.h"
#include "resource.h"

LRESULT CALLBACK WndProc    (HWND, UINT, WPARAM, LPARAM);
BOOL    CALLBACK AboutProc  (HWND, UINT, WPARAM, LPARAM);

void  DrawRectangle (HDC, int, int, DWORD *, DWORD *);
```

```
void  ErrorMessage  (HWND, TCHAR *, TCHAR *) ;
void  DoCaption     (HWND, TCHAR *) ;
int   AskAboutSave  (HWND, TCHAR *) ;

TCHAR * szPerc [NUM_PERC] =
{
 TEXT ("Acoustic Bass Drum"), TEXT ("Bass Drum 1"),
 TEXT ("Side Stick"),         TEXT ("Acoustic Snare"),
 TEXT ("Hand Clap"),          TEXT ("Electric Snare"),
 TEXT ("Low Floor Tom"),      TEXT ("Closed High Hat"),
 TEXT ("High Floor Tom"),     TEXT ("Pedal High Hat"),
 TEXT ("Low Tom"),            TEXT ("Open High Hat"),
 TEXT ("Low - Mid Tom"),      TEXT ("High - Mid Tom"),
 TEXT ("Crash Cymbal 1"),     TEXT ("High Tom"),
 TEXT ("Ride Cymbal 1"),      TEXT ("Chinese Cymbal"),
 TEXT ("Ride Bell"),          TEXT ("Tambourine"),
 TEXT ("Splash Cymbal"),      TEXT ("Cowbell"),
 TEXT ("Crash Cymbal 2"),     TEXT ("Vibraslap"),
 TEXT ("Ride Cymbal 2"),      TEXT ("High Bongo"),
 TEXT ("Low Bongo"),          TEXT ("Mute High Conga"),
 TEXT ("Open High Conga"),    TEXT ("Low Conga"),
 TEXT ("High Timbale"),       TEXT ("Low Timbale"),
 TEXT ("High Agogo"),         TEXT ("Low Agogo"),
 TEXT ("Cabasa"),             TEXT ("Maracas"),
 TEXT ("Short Whistle"),      TEXT ("Long Whistle"),
 TEXT ("Short Guiro"),        TEXT ("Long Guiro"),
 TEXT ("Claves"),             TEXT ("High Wood Block"),
 TEXT ("Low Wood Block"),     TEXT ("Mute Cuica"),
 TEXT ("Open Cuica"),         TEXT ("Mute Triangle"),
 TEXT ("Open Triangle")
} ;

TCHAR  szAppName [] = TEXT ("Drum") ;
TCHAR  szUntitled [] = TEXT ("(Untitled)") ;
TCHAR  szBuffer [80 + MAX_PATH] ;
HANDLE hInst ;
int    cxChar, cyChar ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     hInst = hInstance ;

     wndclass.style          = CS_HREDRAW | CS_VREDRAW ;
```

```
     wndclass.lpfnWndProc   = WndProc ;
     wndclass.cbClsExtra    = 0 ;
     wndclass.cbWndExtra    = 0 ;
     wndclass.hInstance     = hInstance ;
     wndclass.hIcon         = LoadIcon (hInstance, szAppName) ;
     wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
     wndclass.hbrBackground = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName  = szAppName ;
     wndclass.lpszClassName = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, NULL,
                          WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU |
                                    WS_MINIMIZEBOX | WS_HSCROLL | WS_VSCROLL,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, szCmdLine) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
          DispatchMessage (&msg) ;
     }
     return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
     static BOOL     bNeedSave ;
     static DRUM     drum ;
     static HMENU    hMenu ;
     static int      iTempo = 50, iIndexLast ;
     static TCHAR    szFileName [MAX_PATH], szTitleName [MAX_PATH] ;
     HDC             hdc ;
     int             i, x, y ;
     PAINTSTRUCT     ps ;
     POINT           point ;
     RECT            rect ;
```

```
    TCHAR       * szError ;

    switch (message)
    {
    case WM_CREATE:
            // Initialize DRUM structure

        drum.iMsecPerBeat = 100 ;
        drum.iVelocity    = 64 ;
        drum.iNumBeats    = 32 ;

        DrumSetParams (&drum) ;

            // Other initialization

        cxChar = LOWORD (GetDialogBaseUnits ()) ;
        cyChar = HIWORD (GetDialogBaseUnits ()) ;

        GetWindowRect (hwnd, &rect) ;
        MoveWindow (hwnd, rect.left, rect.top,
                          77 * cxChar, 29 * cyChar, FALSE) ;

        hMenu = GetMenu (hwnd) ;

            // Initialize "Volume" scroll bar

        SetScrollRange (hwnd, SB_HORZ, 1, 127, FALSE) ;
        SetScrollPos   (hwnd, SB_HORZ, drum.iVelocity, TRUE) ;

            // Initialize "Tempo" scroll bar

        SetScrollRange (hwnd, SB_VERT, 0, 100, FALSE) ;
        SetScrollPos   (hwnd, SB_VERT, iTempo, TRUE) ;

        DoCaption (hwnd, szTitleName) ;
        return 0 ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDM_FILE_NEW:
            if (bNeedSave && IDCANCEL == AskAboutSave (hwnd, szTitleName))
                return 0 ;

                // Clear drum pattern

            for (i = 0 ; i < NUM_PERC ; i++)
```

```
            {
                drum.dwSeqPerc [i] = 0 ;
                drum.dwSeqPian [i] = 0 ;
            }

            InvalidateRect (hwnd, NULL, FALSE) ;
            DrumSetParams (&drum) ;
            bNeedSave = FALSE ;
            return 0 ;

        case IDM_FILE_OPEN:
                // Save previous file

            if (bNeedSave && IDCANCEL ==
                AskAboutSave (hwnd, szTitleName))
                return 0 ;

                // Open the selected file

            if (DrumFileOpenDlg (hwnd, szFileName, szTitleName))
            {
                szError = DrumFileRead (&drum, szFileName) ;

                if (szError != NULL)
                {
                    ErrorMessage (hwnd, szError, szTitleName) ;
                    szTitleName [0] = '\0' ;
                }
                else
                {
                        // Set new parameters

                    iTempo = (int) (50 *
                        (log10 (drum.iMsecPerBeat) - 1)) ;

                    SetScrollPos (hwnd, SB_VERT, iTempo, TRUE) ;
                    SetScrollPos (hwnd, SB_HORZ, drum.iVelocity, TRUE) ;

                    DrumSetParams (&drum) ;
                    InvalidateRect (hwnd, NULL, FALSE) ;
                    bNeedSave = FALSE ;
                }

                DoCaption (hwnd, szTitleName) ;
            }
            return 0 ;

        case IDM_FILE_SAVE:
```

```
case IDM_FILE_SAVE_AS:
        // Save the selected file

    if ((LOWORD (wParam) == IDM_FILE_SAVE && szTitleName [0]) ||
            DrumFileSaveDlg (hwnd, szFileName, szTitleName))
    {
        szError = DrumFileWrite (&drum, szFileName) ;

        if (szError != NULL)
        {
            ErrorMessage (hwnd, szError, szTitleName) ;
            szTitleName [0] = '\0' ;
        }
        else
            bNeedSave = FALSE ;

        DoCaption (hwnd, szTitleName) ;
    }
    return 0 ;

case IDM_APP_EXIT:
    SendMessage (hwnd, WM_SYSCOMMAND, SC_CLOSE, 0L) ;
    return 0 ;

case IDM_SEQUENCE_RUNNING:
        // Begin sequence

    if (! DrumBeginSequence (hwnd))
    {
        ErrorMessage (hwnd,
            TEXT ("Could not start MIDI sequence --- ")
            TEXT ("MIDI Mapper device is unavailable!"),
            szTitleName) ;
    }
    else
    {
        CheckMenuItem (hMenu, IDM_SEQUENCE_RUNNING,   MF_CHECKED) ;
        CheckMenuItem (hMenu, IDM_SEQUENCE_STOPPED, MF_UNCHECKED) ;
    }
    return 0 ;

case IDM_SEQUENCE_STOPPED:
        // Finish at end of sequence

    DrumEndSequence (FALSE) ;
    return 0 ;
```

```
        case IDM_APP_ABOUT:
            DialogBox (hInst, TEXT ("AboutBox"), hwnd, AboutProc) ;
            return 0 ;
        }
        return 0 ;

    case WM_LBUTTONDOWN:
    case WM_RBUTTONDOWN:
        hdc = GetDC (hwnd) ;

            // Convert mouse coordinates to grid coordinates

        x =      LOWORD (lParam) / cxChar - 40 ;
        y = 2 * HIWORD (lParam) / cyChar - 2 ;
            // Set a new number of beats of sequence

        if (x > 0 && x <= 32 && y < 0)
        {
            SetTextColor (hdc, RGB (255, 255, 255)) ;
            TextOut (hdc, (40 + drum.iNumBeats) * cxChar, 0, TEXT (":!"), 2); ·
            SetTextColor (hdc, RGB (0, 0, 0)) ;

            if (drum.iNumBeats % 4 == 0)
                TextOut (hdc, (40 + drum.iNumBeats) * cxChar, 0,
                         TEXT ("."), 1) ;

            drum.iNumBeats = x ;

            TextOut (hdc, (40 + drum.iNumBeats) * cxChar, 0, TEXT (":!"), 2);

            bNeedSave = TRUE ;
        }

            // Set or reset a percussion instrument beat

        if (x >= 0 && x < 32 && y >= 0 && y < NUM_PERC)
        {
            if (message == WM_LBUTTONDOWN)
                drum.dwSeqPerc[y] ^= (1 << x) ;
            else
                drum.dwSeqPian[y] ^= (1 << x) ;

            DrawRectangle (hdc, x, y, drum.dwSeqPerc, drum.dwSeqPian) ;

            bNeedSave = TRUE ;
        }

        ReleaseDC (hwnd, hdc) ;
```

```
            DrumSetParams (&drum) ;
            return 0 ;

        case WM_HSCROLL:
                // Change the note velocity

            switch (LOWORD (wParam))
            {
            case SB_LINEUP:         drum.iVelocity -= 1 ;   break ;
            case SB_LINEDOWN:       drum.iVelocity += 1 ;   break ;
            case SB_PAGEUP:         drum.iVelocity -= 8 ;   break ;
            case SB_PAGEDOWN:       drum.iVelocity += 8 ;   break ;
            case SB_THUMBPOSITION:
                drum.iVelocity = HIWORD (wParam) ;
                break ;

            default:
                return 0 ;
            }

            drum.iVelocity = max (1, min (drum.iVelocity, 127)) ;
            SetScrollPos (hwnd, SB_HORZ, drum.iVelocity, TRUE) ;
            DrumSetParams (&drum) ;
            bNeedSave = TRUE ;
            return 0 ;

        case WM_VSCROLL:
                // Change the tempo

            switch (LOWORD (wParam))
            {
            case SB_LINEUP:         iTempo -=  1 ;   break ;
            case SB_LINEDOWN:       iTempo +=  1 ;   break ;
            case SB_PAGEUP:         iTempo -= 10 ;   break ;
            case SB_PAGEDOWN:       iTempo += 10 ;   break ;
            case SB_THUMBPOSITION:
                iTempo = HIWORD (wParam) ;
                break ;

            default:
                return 0 ;
            }

            iTempo = max (0, min (iTempo, 100)) ;
            SetScrollPos (hwnd, SB_VERT, iTempo, TRUE) ;

            drum.iMsecPerBeat = (WORD) (10 * pow (100, iTempo / 100.0)) ;
```

```
            DrumSetParams (&drum) ;
            bNeedSave = TRUE ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;

            SetTextAlign (hdc, TA_UPDATECP) ;
            SetBkMode (hdc, TRANSPARENT) ;

                // Draw the text strings and horizontal lines

            for (i = 0 ; i < NUM_PERC ; i++)
            {
                MoveToEx (hdc,i & 1 ? 20 * cxChar : cxChar,
                          (2 * i + 3) * cyChar / 4, NULL) ;

                TextOut (hdc, 0, 0, szPerc [i], lstrlen (szPerc [i])) ;

                GetCurrentPositionEx (hdc, &point) ;

                MoveToEx (hdc, point.x + cxChar, point.y + cyChar / 2, NULL) ;
                LineTo   (hdc,        39 * cxChar, point.y + cyChar / 2) ;
            }

            SetTextAlign (hdc, 0) ;

                // Draw rectangular grid, repeat mark, and beat marks

            for (x = 0 ; x < 32 ; x++)
            {
                for (y = 0 ; y < NUM_PERC ; y++)
                    DrawRectangle (hdc, x, y, drum.dwSeqPerc, drum.dwSeqPian) ;

                SetTextColor (hdc, x == drum.iNumBeats - 1 ?
                                   RGB (0, 0, 0) : RGB (255, 255, 255)) ;

                TextOut (hdc, (41 + x) * cxChar, 0, TEXT (":!"), 2) ;

                SetTextColor (hdc, RGB (0, 0, 0)) ;

                if (x % 4 == 0)
                    TextOut (hdc, (40 + x) * cxChar, 0, TEXT ("."), 1) ;
            }

            EndPaint (hwnd, &ps) ;
            return 0 ;
```

```
case WM_USER_NOTIFY:
        // Draw the "bouncing ball"

    hdc = GetDC (hwnd) ;

    SelectObject (hdc, GetStockObject (NULL_PEN)) ;
    SelectObject (hdc, GetStockObject (WHITE_BRUSH)) ;

    for (i = 0 ; i < 2 ; i++)
    {
        x = iIndexLast ;
        y = NUM_PERC + 1 ;

        Ellipse (hdc, (x + 40) * cxChar, (2 * y + 3) * cyChar / 4,
           (x + 41) * cxChar, (2 * y + 5) * cyChar / 4);

        iIndexLast = wParam ;
        SelectObject (hdc, GetStockObject (BLACK_BRUSH)) ;
    }

    ReleaseDC (hwnd, hdc) ;
    return 0 ;

case WM_USER_ERROR:
    ErrorMessage (hwnd, TEXT ("Can't set timer event for tempo"),
                  szTitleName) ;

                                    // fall through
case WM_USER_FINISHED:
    DrumEndSequence (TRUE) ;
    CheckMenuItem (hMenu, IDM_SEQUENCE_RUNNING, MF_UNCHECKED) ;
    CheckMenuItem (hMenu, IDM_SEQUENCE_STOPPED, MF_CHECKED) ;
    return 0 ;

case WM_CLOSE:
    if (!bNeedSave || IDCANCEL != AskAboutSave (hwnd, szTitleName))
        DestroyWindow (hwnd) ;

    return 0 ;

case WM_QUERYENDSESSION:
    if (!bNeedSave || IDCANCEL != AskAboutSave (hwnd, szTitleName))
        return 1L ;

    return 0 ;
```

```
        case WM_DESTROY:
            DrumEndSequence (TRUE) ;
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK AboutProc (HWND hDlg, UINT message, WPARAM wParam, LPARAM lParam)
{
        switch (message)
        {
        case WM_INITDIALOG:
            return TRUE ;

        case WM_COMMAND:
            switch (LOWORD (wParam))
            {
            case IDOK:
                EndDialog (hDlg, 0) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}

void DrawRectangle (HDC hdc, int x, int y, DWORD * dwSeqPerc, DWORD * dwSeqPian)
{
        int iBrush ;

        if (dwSeqPerc [y] & dwSeqPian [y] & (1L << x))
            iBrush = BLACK_BRUSH ;

        else if (dwSeqPerc [y] & (1L << x))
            iBrush = DKGRAY_BRUSH ;

        else if (dwSeqPian [y] & (1L << x))
            iBrush = LTGRAY_BRUSH ;

        else
            iBrush = WHITE_BRUSH ;

        SelectObject (hdc, GetStockObject (iBrush)) ;

        Rectangle (hdc, (x + 40) * cxChar     , (2 * y + 4) * cyChar / 4,
                        (x + 41) * cxChar + 1, (2 * y + 6) * cyChar / 4 + 1) ;
}
```

```
void ErrorMessage (HWND hwnd, TCHAR * szError, TCHAR * szTitleName)
{
     wsprintf (szBuffer, szError,
          (LPSTR) (szTitleName [0] ? szTitleName : szUntitled)) ;

     MessageBeep (MB_ICONEXCLAMATION) ;
     MessageBox (hwnd, szBuffer, szAppName, MB_OK | MB_ICONEXCLAMATION) ;
}

void DoCaption (HWND hwnd, TCHAR * szTitleName)
{
     wsprintf (szBuffer, TEXT ("MIDI Drum Machine - %s"),
          (LPSTR) (szTitleName [0] ? szTitleName : szUntitled)) ;

     SetWindowText (hwnd, szBuffer) ;
}

int AskAboutSave (HWND hwnd, TCHAR * szTitleName)
{
     int iReturn ;

     wsprintf (szBuffer, TEXT ("Save current changes in %s?"),
          (LPSTR) (szTitleName [0] ? szTitleName : szUntitled)) ;

     iReturn = MessageBox (hwnd, szBuffer, szAppName,
                           MB_YESNOCANCEL | MB_ICONQUESTION) ;

     if (iReturn == IDYES)
          if (!SendMessage (hwnd, WM_COMMAND, IDM_FILE_SAVE, 0))
               iReturn = IDCANCEL ;

     return iReturn ;
}
```

DRUMTIME.H

```
/*-----------------------------------------
     DRUMTIME.H Header File for Time Functions for DRUM Program
   -----------------------------------------*/

#define NUM_PERC              47
#define WM_USER_NOTIFY        (WM_USER + 1)
#define WM_USER_FINISHED      (WM_USER + 2)
#define WM_USER_ERROR         (WM_USER + 3)

#pragma pack(push, 2)
```

```
typedef struct
{
      short   iMsecPerBeat ;
      short   iVelocity ;
      short   iNumBeats ;
      DWORD dwSeqPerc [NUM_PERC] ;
      DWORD dwSeqPian [NUM_PERC] ;
}
DRUM, * PDRUM ;

# pragma pack(pop)

void DrumSetParams        (PDRUM) ;
BOOL DrumBeginSequence   (HWND)  ;
void DrumEndSequence      (BOOL)  ;
```

DRUMTIME.C

```
/* ---------------------
      DRUMFILE.C——Timer Routines for DRUM
                    (c) Charles Petzold, 1998
   ---------------------*/

# include <windows.h>
# include "drumtime.h"

# define minmax(a,x,b) (min (max (x, a), b))

# define TIMER_RES      5

void CALLBACK DrumTimerFunc (UINT, UINT, DWORD, DWORD, DWORD) ;

BOOL        bSequenceGoing, bEndSequence ;
DRUM        drum ;
HMIDIOUT    hMidiOut ;
HWND        hwndNotify ;
int         iIndex ;
UINT        uTimerRes, uTimerID ;

DWORD MidiOutMessage (HMIDIOUT hMidi, int iStatus, int iChannel,
                      int iData1, int iData2)
{
      DWORD dwMessage ;

      dwMessage = iStatus | iChannel | (iData1 << 8) | (iData2 << 16) ;

      return midiOutShortMsg (hMidi, dwMessage) ;
}
```

```
void DrumSetParams (PDRUM pdrum)
{
     CopyMemory (&drum, pdrum, sizeof (DRUM)) ;
}

BOOL DrumBeginSequence (HWND hwnd)
{
     TIMECAPS tc ;

     hwndNotify = hwnd ;          // Save window handle for notification
     DrumEndSequence (TRUE) ;     // Stop current sequence if running

          // Open the MIDI Mapper output port

     if (midiOutOpen (&hMidiOut, MIDIMAPPER, 0, 0, 0))
          return FALSE ;

          // Send Program Change messages for channels 9 and 0

     MidiOutMessage (hMidiOut, 0xC0, 9, 0, 0) ;
     MidiOutMessage (hMidiOut, 0xC0, 0, 0, 0) ;

          // Begin sequence by setting a timer event

     timeGetDevCaps (&tc, sizeof (TIMECAPS)) ;
     uTimerRes = minmax (tc.wPeriodMin, TIMER_RES, tc.wPeriodMax) ;
     timeBeginPeriod (uTimerRes) ;

     uTimerID = timeSetEvent (max ((UINT) uTimerRes, (UINT) drum.iMsecPerBeat),
                              uTimerRes, DrumTimerFunc, 0, TIME_ONESHOT) ;

     if (uTimerID == 0)
     {
          timeEndPeriod (uTimerRes) ;
          midiOutClose (hMidiOut) ;
          return FALSE ;
     }

     iIndex = -1 ;
     bEndSequence = FALSE ;
     bSequenceGoing = TRUE ;

     return TRUE ;
}

void DrumEndSequence (BOOL bRightAway)
{
     if (bRightAway)
```

```
    {
        if (bSequenceGoing)
        {
                // stop the timer
            if (uTimerID)
                timeKillEvent (uTimerID) ;
            timeEndPeriod (uTimerRes) ;

                // turn off all notes
            MidiOutMessage (hMidiOut, 0xB0, 9, 123, 0) ;
            MidiOutMessage (hMidiOut, 0xB0, 0, 123, 0) ;

                // close the MIDI port
            midiOutClose (hMidiOut) ;
            bSequenceGoing = FALSE ;
        }
    }
    else
        bEndSequence = TRUE ;
}

void CALLBACK DrumTimerFunc (UINT uID, UINT uMsg, DWORD dwUser,
                             DWORD dw1, DWORD dw2)
{
    static DWORD dwSeqPercLast [NUM_PERC], dwSeqPianLast [NUM_PERC] ;
    int          i ;

        // Note Off messages for channels 9 and 0

    if (iIndex != -1)
    {
        for (i = 0 ; i < NUM_PERC ; i++)
        {
            if (dwSeqPercLast[i] & 1 << iIndex)
                MidiOutMessage (hMidiOut, 0x80, 9, i + 35, 0) ;

            if (dwSeqPianLast[i] & 1 << iIndex)
                MidiOutMessage (hMidiOut, 0x80, 0, i + 35, 0) ;
        }
    }

        // Increment index and notify window to advance bouncing ball

    iIndex = (iIndex + 1) % drum.iNumBeats ;
    PostMessage (hwndNotify, WM_USER_NOTIFY, iIndex, timeGetTime ()) ;

        // Check if ending the sequence
```

```
    if (bEndSequence && iIndex == 0)
    {
        PostMessage (hwndNotify, WM_USER_FINISHED, 0, 0L) ;
        return ;
    }

        // Note On messages for channels 9 and 0

    for (i = 0 ; i < NUM_PERC ; i++)
    {
        if (drum.dwSeqPerc[i] & 1 << iIndex)
            MidiOutMessage (hMidiOut, 0x90, 9, i + 35, drum.iVelocity) ;

        if (drum.dwSeqPian[i] & 1 << iIndex)
            MidiOutMessage (hMidiOut, 0x90, 0, i + 35, drum.iVelocity) ;

        dwSeqPercLast[i] = drum.dwSeqPerc[i] ;
        dwSeqPianLast[i] = drum.dwSeqPian[i] ;
    }
        // Set a new timer event

    uTimerID = timeSetEvent (max ((int) uTimerRes, drum.iMsecPerBeat),
                             uTimerRes, DrumTimerFunc, 0, TIME_ONESHOT) ;

    if (uTimerID == 0)
    {
        PostMessage (hwndNotify, WM_USER_ERROR, 0, 0) ;
    }
}
```

DRUMFILE.H

```
/*-------------------------------------
    DRUMFILE.H Header File for File I/O Routines for DRUM
  -------------------------------------*/

BOOL    DrumFileOpenDlg (HWND, TCHAR *, TCHAR *) ;
BOOL    DrumFileSaveDlg (HWND, TCHAR *, TCHAR *) ;
TCHAR * DrumFileWrite   (DRUM *, TCHAR *) ;
TCHAR * DrumFileRead    (DRUM *, TCHAR *) ;
```

DRUMFILE.C

```
/*-------------------------------------
    DRUMFILE.C——File I/O Routines for DRUM
                (c) Charles Petzold, 1998
  -------------------------------------*/

#include <windows.h>
```

```
# include <commdlg.h>
# include "drumtime.h"
# include "drumfile.h"

OPENFILENAME ofn = { sizeof (OPENFILENAME) } ;

TCHAR * szFilter [] = { TEXT ("Drum Files ( * .DRM)"),
                        TEXT ("* .drm"), TEXT ("") } ;

TCHAR szDrumID   [] = TEXT ("DRUM") ;
TCHAR szListID   [] = TEXT ("LIST") ;
TCHAR szInfoID   [] = TEXT ("INFO") ;
TCHAR szSoftID   [] = TEXT ("ISFT") ;
TCHAR szDateID   [] = TEXT ("ISCD") ;
TCHAR szFmtID    [] = TEXT ("fmt ") ;
TCHAR szDataID   [] = TEXT ("data") ;
char szSoftware  [] = "DRUM by Charles Petzold, Programming Windows" ;

TCHAR szErrorNoCreate       [] = TEXT ("File %s could not be opened for writing.");
TCHAR szErrorCannotWrite    [] = TEXT ("File %s could not be written to. ") ;
TCHAR szErrorNotFound       [] = TEXT ("File %s not found or cannot be opened.") ;
TCHAR szErrorNotDrum        [] = TEXT ("File %s is not a standard DRUM file.") ;
TCHAR szErrorUnsupported    [] = TEXT ("File %s is not a supported DRUM file.") ;
TCHAR szErrorCannotRead     [] = TEXT ("File %s cannot be read.") ;

BOOL DrumFileOpenDlg (HWND hwnd, TCHAR * szFileName, TCHAR * szTitleName)
{
     ofn.hwndOwner         = hwnd ;
     ofn.lpstrFilter       = szFilter [0] ;
     ofn.lpstrFile         = szFileName ;
     ofn.nMaxFile          = MAX_PATH ;
     ofn.lpstrFileTitle    = szTitleName ;
     ofn.nMaxFileTitle     = MAX_PATH ;
     ofn.Flags             = OFN_CREATEPROMPT ;
     ofn.lpstrDefExt       = TEXT ("drm") ;

     return GetOpenFileName (&ofn) ;
}
BOOL DrumFileSaveDlg (HWND hwnd, TCHAR * szFileName, TCHAR * szTitleName)
{
     ofn.hwndOwner         = hwnd ;
     ofn.lpstrFilter       = szFilter [0] ;
     ofn.lpstrFile         = szFileName ;
     ofn.nMaxFile          = MAX_PATH ;
     ofn.lpstrFileTitle    = szTitleName ;
     ofn.nMaxFileTitle     = MAX_PATH ;
     ofn.Flags             = OFN_OVERWRITEPROMPT ;
     ofn.lpstrDefExt       = TEXT ("drm") ;
```

```
    return GetSaveFileName (&ofn) ;
}

TCHAR * DrumFileWrite (DRUM * pdrum, TCHAR * szFileName)
{
    char            szDateBuf [16] ;
    HMMIO           hmmio ;
    int             iFormat = 2 ;
    MMCKINFO        mmckinfo [3] ;
    SYSTEMTIME      st ;
    WORD            wError = 0 ;

    memset (mmckinfo, 0, 3 * sizeof (MMCKINFO)) ;

        // Recreate the file for writing

    if ((hmmio = mmioOpen (szFileName, NULL,
            MMIO_CREATE | MMIO_WRITE | MMIO_ALLOCBUF)) == NULL)
        return szErrorNoCreate ;

            // Create a "RIFF" chunk with a "CPDR" type

    mmckinfo[0].fccType = mmioStringToFOURCC (szDrumID, 0) ;

    wError |= mmioCreateChunk (hmmio, &mmckinfo[0], MMIO_CREATERIFF) ;

        // Create "LIST" sub-chunk with an "INFO" type

    mmckinfo[1].fccType = mmioStringToFOURCC (szInfoID, 0) ;

    wError |= mmioCreateChunk (hmmio, &mmckinfo[1], MMIO_CREATELIST) ;

        // Create "ISFT" sub-sub-chunk

    mmckinfo[2].ckid = mmioStringToFOURCC (szSoftID, 0) ;

    wError |= mmioCreateChunk (hmmio, &mmckinfo[2], 0) ;
    wError |= (mmioWrite (hmmio, szSoftware, sizeof (szSoftware)) !=
                                        sizeof (szSoftware)) ;
    wError |= mmioAscend (hmmio, &mmckinfo[2], 0) ;

        // Create a time string
    GetLocalTime (&st) ;

    wsprintfA (szDateBuf, "%04d-%02d-%02d", st.wYear, st.wMonth, st.wDay) ;

        // Create "ISCD" sub-sub-chunk
```

```
    mmckinfo[2].ckid = mmioStringToFOURCC (szDateID, 0) ;

    wError |= mmioCreateChunk (hmmio, &mmckinfo[2], 0) ;
    wError |= (mmioWrite (hmmio, szDateBuf, (strlen (szDateBuf) + 1)) !=
                                    (int) (strlen (szDateBuf) + 1)) ;
    wError |= mmioAscend (hmmio, &mmckinfo[2], 0) ;
    wError |= mmioAscend (hmmio, &mmckinfo[1], 0) ;

        // Create "fmt " sub-chunk

    mmckinfo[1].ckid = mmioStringToFOURCC (szFmtID, 0) ;

    wError |= mmioCreateChunk (hmmio, &mmckinfo[1], 0) ;
    wError |= (mmioWrite (hmmio, (PSTR) &iFormat, sizeof (int)) !=
                                                sizeof (int)) ;
    wError |= mmioAscend (hmmio, &mmckinfo[1], 0) ;

        // Create the "data" sub-chunk

    mmckinfo[1].ckid = mmioStringToFOURCC (szDataID, 0) ;

    wError |= mmioCreateChunk (hmmio, &mmckinfo[1], 0) ;
    wError |= (mmioWrite (hmmio, (PSTR) pdrum, sizeof (DRUM)) !=
                                            sizeof (DRUM)) ;
    wError |= mmioAscend (hmmio, &mmckinfo[1], 0) ;
    wError |= mmioAscend (hmmio, &mmckinfo[0], 0) ;

        // Clean up and return

    wError |= mmioClose (hmmio, 0) ;

    if (wError)
    {
        mmioOpen (szFileName, NULL, MMIO_DELETE) ;
        return szErrorCannotWrite ;
    }
    return NULL ;
}

TCHAR * DrumFileRead (DRUM * pdrum, TCHAR * szFileName)
{
    DRUM        drum ;
    HMMIO       hmmio ;
    int         i, iFormat ;
    MMCKINFO    mmckinfo [3] ;

    ZeroMemory (mmckinfo, 2 * sizeof (MMCKINFO)) ;
```

```
    // Open the file

if ((hmmio = mmioOpen (szFileName, NULL, MMIO_READ)) == NULL)
    return szErrorNotFound ;

    // Locate a "RIFF" chunk with a "DRUM" form-type

mmckinfo[0].ckid = mmioStringToFOURCC (szDrumID, 0) ;

if (mmioDescend (hmmio, &mmckinfo[0], NULL, MMIO_FINDRIFF))
{
     mmioClose (hmmio, 0) ;
     return szErrorNotDrum ;
}

    // Locate, read, and verify the "fmt " sub-chunk

mmckinfo[1].ckid = mmioStringToFOURCC (szFmtID, 0) ;

if (mmioDescend (hmmio, &mmckinfo[1], &mmckinfo[0], MMIO_FINDCHUNK))
{
     mmioClose (hmmio, 0) ;
     return szErrorNotDrum ;
}

if (mmckinfo[1].cksize != sizeof (int))
{
     mmioClose (hmmio, 0) ;
     return szErrorUnsupported ;
}

if (mmioRead (hmmio, (PSTR) &iFormat, sizeof (int)) != sizeof (int))
{
     mmioClose (hmmio, 0) ;
     return szErrorCannotRead ;
}

if (iFormat != 1 && iFormat != 2)
{
     mmioClose (hmmio, 0) ;
     return szErrorUnsupported ;
}

    // Go to end of "fmt " sub-chunk

mmioAscend (hmmio, &mmckinfo[1], 0) ;

    // Locate, read, and verify the "data" sub-chunk
```

```
    mmckinfo[1].ckid = mmioStringToFOURCC (szDataID, 0) ;

    if (mmioDescend (hmmio, &mmckinfo[1], &mmckinfo[0], MMIO_FINDCHUNK))
    {
         mmioClose (hmmio, 0) ;
         return szErrorNotDrum ;
    }

    if (mmckinfo[1].cksize != sizeof (DRUM))
    {
         mmioClose (hmmio, 0) ;
         return szErrorUnsupported ;
    }

    if (mmioRead (hmmio, (LPSTR) &drum, sizeof (DRUM)) != sizeof (DRUM))
    {
         mmioClose (hmmio, 0) ;
         return szErrorCannotRead ;
    }

        // Close the file

    mmioClose (hmmio, 0) ;

        // Convert format 1 to format 2 and copy the DRUM structure data

    if (iFormat == 1)
    {
         for (i = 0 ; i < NUM_PERC ; i++)
         {
              drum.dwSeqPerc [i] = drum.dwSeqPian [i] ;
              drum.dwSeqPian [i] = 0 ;
         }
    }

    memcpy (pdrum, &drum, sizeof (DRUM)) ;
    return NULL ;
}
```

DRUM.RC（摘录）

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Menu
```

```
DRUM MENU DISCARDABLE
BEGIN
    POPUP "&File"
    BEGIN
        MENUITEM "&New",                    IDM_FILE_NEW
        MENUITEM "&Open...",                IDM_FILE_OPEN
        MENUITEM "&Save",                   IDM_FILE_SAVE
        MENUITEM "Save &As...",             IDM_FILE_SAVE_AS
        MENUITEM SEPARATOR
        MENUITEM "E&xit",                   IDM_APP_EXIT
    END
    POPUP "&Sequence"
    BEGIN
        MENUITEM "&Running",                IDM_SEQUENCE_RUNNING
        MENUITEM "&Stopped",                IDM_SEQUENCE_STOPPED
        , CHECKED
    END
    POPUP "&Help"
    BEGIN
        MENUITEM "&About...",               IDM_APP_ABOUT
    END
END

//////////////////////////////////////////////////////////////////////////////
// Icon

DRUM                    ICON        DISCARDABLE         "drum.ico"

//////////////////////////////////////////////////////////////////////////////
// Dialog

ABOUTBOX DIALOG DISCARDABLE 20, 20, 160, 164
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON   "OK",IDOK,54,143,50,14
    ICON            "DRUM",IDC_STATIC,8,8,21,20
    CTEXT           "DRUM",IDC_STATIC,34,12,90,8
    CTEXT           "MIDI Drum Machine",IDC_STATIC,7,36,144,8
    CONTROL         "",IDC_STATIC,"Static",SS_BLACKFRAME,8,88,144,46
    LTEXT           "Left Button:\t\tDrum sounds",IDC_STATIC,12,92,136,8
    LTEXT           "Right Button:\t\tPiano sounds",IDC_STATIC,12,102,136,8
    LTEXT           "Horizontal Scroll:\t\tVelocity",IDC_STATIC,12,112,136,8
    LTEXT           "Vertical Scroll:\t\tTempo",IDC_STATIC,12,122,136,8
    CTEXT           "Copyright (c) Charles Petzold, 1998",IDC_STATIC,8,48,
```

```
                144,8
    CTEXT       "" "Programming Windows," " 5th Edition",IDC_STATIC,8,60,
                144,8
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by Drum.rc

#define IDM_FILE_NEW                40001
#define IDM_FILE_OPEN               40002
#define IDM_FILE_SAVE               40003
#define IDM_FILE_SAVE_AS            40004
#define IDM_APP_EXIT                40005
#define IDM_SEQUENCE_RUNNING        40006
#define IDM_SEQUENCE_STOPPED        40007
#define IDM_APP_ABOUT               40008
```

当第一次运行 DRUM 时，您将看到在窗口中有两列，左边一列按名称列出了 47 种不同的打击乐器；右边的网格是打击乐器的声音与时间的二维数组。每一个打击器都对应网格中的一行，32 列就是 32 拍。如果要让这 32 拍出现在一个 4/4 拍的小节中(即每小节 4 个四分音符)，那么每 1 拍对应一个三十二分音符。

从 Sequence 菜单选择 Running 时，程序将试图打开 MIDI Mapper 设备。如果失败，屏幕将出现一个消息框。否则，您将看到一个“跳动的小球”随演奏的节拍在网格底部跳过。

在网格的任何位置单击鼠标左键都可以在此拍中演奏打击乐器的声音，这时区域将变成暗灰色。用鼠标右键还可以添加钢琴的拍子，这时区域将会变成浅灰色。如果按下两个键(同时或分别)，此区域将变成黑色，而且可以同时听到打击乐器和钢琴的声音。再次单击其中的一个键或双键，将关闭该拍中的声音。

网格上部是每 4 拍一个点。这些点使我们不用过多的计算就可以很简易地确定单击的位置。网格的右上角是一个冒号和一条竖线(:|)，它们看起来像传统音乐符号中的反复记号。这个符号表示序列的长度。您可以通过单击鼠标来将反复记号放置于网格内的任意位置。该序列最多(但不包括)只能演奏反复记号以内的拍子。如果要创建华尔兹节奏，则应将反复记号设定为 3 拍的若干倍。

水平滚动条控制 MIDI Note On 消息中的速率字节。这虽然能改变一些合成器的音质，但一般会影响音量。程序起初将速率滚动条设置在中间位置。竖直滚动条控制拍子。这是对数刻度，范围从每拍 1 秒(滚动条在底部)到每拍 10 毫秒(滚动条在顶部)。程序最初将拍子设置为每拍 100 毫秒(1/10 秒)，这时滚动条在中间。

File 菜单允许您保存和读取扩展名为 .DRM 的文件，这是我定义的一种格式。这些

文件很小,并采用了 RIFF 的文件格式,这是一种所有新的多媒体数据文件推荐使用的格式。Help 菜单中的 About 选项显示一个对话框,该对话框用一段非常简明的摘要来说明鼠标在网格中的用法以及两个滚动条的功能。

最后,Sequence 菜单中的 Stopped 选项用于当前序列结束后终止乐曲并关闭 MIDI Mapper 设备。

22.3.8　多媒体 time 函数

您可能会注意到 DRUM.C 没有调用任何多媒体函数,而所有的实际操作都发生在 DRUMTIME 模块中。

虽然普通的 Windows 计时器使用起来很简单,但它对临界时间的应用程序却有灾难性的影响。就像我们在 BACHTOCC 程序中所看到的一样,演奏音乐就是这样的一种临界时间应用程序,对此 Windows 计时器是不合适的。为了提供在 PC 上演奏 MIDI 所需要的精度,多媒体 API 还包括一个高分辨率的计时器,此计时器通过 7 个前缀是 time 的函数实现。这些函数有一个是多余的,而且 DRUMTIME 演示了其余 6 个函数的用途。计时器函数将处理运行在一个单独线程中的回调函数。系统将按照程序指定的计时器延期时间来调用计时器。

处理多媒体计时器时,可以用毫秒指定两种不同的时间。第一个是延期时间,第二个称为分辨度。您可以认为分辨度是容错误差。如果指定一个延期 100 毫秒,而分辨率是 10 毫秒,则计时器的实际延期范围在 90 ~ 110 毫秒之间。

使用计时器之前,应获得计时器的设备能力:

```
timeGetDevCaps (&timecaps, uSize) ;
```

第一个参数是 TIMECAPS 类型结构的指针,第二个参数是此结构的大小。TIMECAPS 结构只有两个字段,wPeriodMin 和 wPeriodMax。这是计时器设备驱动程序所支持的最小和最大的分辨率值。如果调用 timeGetDevCaps 后再查看这些值,会发现 wPeriodMin 是 1 而 wPeriodMax 是 65 535,所以此函数并不很重要。不过,得到这些分辨率值并用于其他计时器函数调用是个好主意。

下一步调用

```
timeBeginPeriod (uResolution) ;
```

以指出程序所需要的计时器分辨率的最低值。该值应在 TIMECAPS 结构所确定的范围之内。此调用允许为可能使用计时器的多个程序提供最好的计时器设备驱动程序。调用 timeBeginPeriod 及 timeEndPeriod 必须成对出现,我将在后面对 timeEndPeriod 作简短的描述。

现在可以真正设置一个计时器事件:

```
idTimer = timeSetEvent (uDelay, uResolution, CallBackFunc, dwData, uFlag) ;
```

如果发生错误,从调用返回的 idTimer 将是 0。在调用的下面,将从 Windows 里用 uDelay 毫秒来调用 CallBackFunc 函数,其中允许的误差由 uResolution 指定。uResolution 值必须大于或等于传递给 timeBeginPeriod 的分辨率。dwData 是程序定义的数据,后来传递给 CallBackFunc。最后一个参数可以是 TIME_ONESHOT,也可以是 TIME_PERIODIC。前者用于在 uDelay 毫秒数中获得一次 CallBackFunc 调用,而后者用于每个 uDelay 毫秒都获得一次 CallBackFunc 调用。

要在调用 CallBackFunc 之前终止只发生一次的计时器事件,或者暂停周期性的计时器事件,请调用

```
timeKillEvent (idTimer) ;
```

调用 CallBackFunc 后不必删除只发生一次的计时器事件。在程序中用完计时器以后,请调用

```
timeEndPeriod (wResolution) ;
```

其中的参数与传递给 timeBeginPeriod 的相同。

另两个函数的前缀是 time。函数

```
dwSysTime = timeGetTime () ;
```

返回从 Windows 第一次启动到现在的系统时间,单位是毫秒。函数

```
timeGetSystemTime (&mmtime, uSize) ;
```

需要一个 MMTIME 结构的指针(与第一个参数一样),以及此结构的大小(与第二个参数一样)。虽然 MMTIME 结构可以在其他环境中用来得到非毫秒格式的系统时间,但此例中它都返回毫秒时间。所以 timeGetSystemTime 是多余的。

回调函数只限于它所能做的 Windows 函数调用中。回调函数可以调用 PostMessage,PostMessage 包含有 4 个计时器函数(timeSetEvent、timeKillEvent、timeGetTime 和多余的 timeGetSystemTime)、两个 MIDI 输出函数(midiOutShortMsg 和 midiOutLongMsg),以及调试函数 OutputDebugStr。

很明显,设计多媒体计时器主要是用于 MIDI 序列而很少用于其他方面。当然,可以使用 PostMessage 来通知计时器事件的窗口过程,而且窗口过程可以做任何它想做的事,只是不能响应计时器回调自身的准确性。

回调函数有 5 个参数,但只使用了其中两个参数:从 timeSetEvent 返回的计时器 ID 号和最初作为参数传递给 timeSetEvent 的 dwData 值。

DRUM.C 模块调用 DRUMTIME.C 中的 DrumSetParams 函数有很多次——创建 DRUM 窗口时、用户在网格上单击或者滚动滚动条时、从磁盘上加载.DRM 文件时,以

及清除网格时。DrumSetParams 的唯一的参数是指向 DRUM 类型结构的指针，此结构类型在 DRUMTIME.H 定义。该结构以毫秒为单位存储拍子时间、速度（通常对应于音量）、序列中的拍数，以及用于存储网格（为打击乐器和钢琴声设定）的两套 47 个 32 字节的整数。这些 32 位整数中的每一位都对应序列的一拍。DRUM.C 模块将在静态内存中维护一个 DRUM 类型的结构，并在调用 DrumSetParams 时向它传递一个指针。DrumSetParams 只简单地复制此结构的内容。

要启动序列，DRUM 调用 DRUMTIME 中的 DrumBeginSequence 函数。唯一的参数就是窗口句柄，其作用是通知。DrumBeginSequence 打开 MIDI Mapper 输出设备，如果成功，则发送 Program Change 消息来为 MIDI 通道 0 和 9 选择乐器声音。（这些通道是基于 0 的，所以 9 实际指的是 MIDI 通道 10，即打击乐器通道，另一个通道用于钢琴声。）DrumBeginSequence 通过调用 timeGetDevCaps 和 timeBeginPeriod 来继续。在 TIMER_RES 定义的理想计时器分辨率通常是 5 毫秒，但我定义了一个被称为“minmax”的宏来计算从 timeGetDevCaps 返回的限制范围以内的分辨率。

下一个调用是 timeSetEvent，用于确定拍子时间，计算分辨率、回调函数 DrumTimerFunc 及 TIME_ONESHOT 常数。DRUMTIME 用的是只有一次的计时器，而不是周期性计时器，所以速度可以随序列的运行而动态变化。timeSetEvent 调用之后，计时器设备驱动程序将在延期时间结束以后调用 DrumTimerFunc。

DrumTimerFunc 回调是 DRUMTIME.C 中的函数，在 DRUMTIME.C 中有许多重要的操作。变量 iIndex 存储序列中当前的拍子。回调从为当前演奏的声音发送 MIDI Note Off 消息开始。iIndex 的初始值 -1 防止第一次启动序列时发生这种情况。

接下来，iIndex 递增并将其值连同用户定义的一个 WM_USER_NOTIFY 消息一起传递给 DRUM 中的窗口句柄。wParam 消息参数设置为 iIndex，以便在 DRUM.C 中，WndProc 能够移动网格底部的“跳动的小球”。

DrumTimerFunc 将下列事件作为结束：把 Note On 消息发送给通道 0 和 9 的合成器上，并保存网格值以便下一次可以关闭声音，然后通过调用 timeSetEvent 来设置新的只发生一次的计时器事件。

要停止序列，DRUM 调用 DrumEndSequence，其中唯一的参数可以设置为 TRUE 或 FALSE。如果是 TRUE，则 DrumEndSequence 按下面的过程立即结束序列：删除所有即将发生的计时器事件来，调用 timeEndPeriod，向两个 MIDI 通道发送 all notes off 消息，然后关闭 MIDI 输出端口。当用户决定终止程序时，DRUM 用 TRUE 参数调用 DrumEndSequence。

然而，当用户在 DRUM 里的 Sequence 菜单中选择 Stop 时，程序将用 FALSE 作为参数调用 DrumEndSequence。这就允许序列在结束之前完成当前的循环。DrumEndSequence 通过把 bEndSequence 全局变量设置为 NULL 来响应此调用。如果 bEndSequence

是 TRUE，并且拍子的索引值设置为 0，则 DrumTimerFunc 把用户定义的 WM_USER_FINISHED 消息发送给 WndProc。WndProc 必须通过用 TRUE 作为参数调用 DrumEndSequence 来响应该消息，以便正确地结束计时器和 MIDI 端口的使用。

22.3.9 RIFF 文件 I/O

DRUM 程序也可以保存和检索包含存储在 DRUM 结构中信息的文件。这些文件格式都是 RIFF(Resource Interchange File Format：资源交换文件格式)，即推荐的多媒体文件类型。当然，您可以用标准文件 I/O 函数来读写 RIFF 文件，但更简便的方法是使用前缀为 mmio(对“多媒体输入/输出”)的函数。

检查 .WAV 格式时我们发现，RIFF 是标记文件格式，这意味着文件中的数据由不同长度的块组成(称为“块”)，每个块都用一个标记来识别。一个标记就是一个 4 字节的 ASCII 字符串，这与 32 位整数的标记名称相比要容易些。标记的后面是块长度及其数据。因为文件中的信息不是位于文件开头固定的偏移量而是用标记定义，所以标记文件格式是通用的。这样，可以通过添加附加标记来增强文件格式。在读文件时，程序可以很容易地找到所需要的数据，并跳过不需要的或者不理解的标记。

Windows 中的 RIFF 文件由独立的块组成，这些块是文件中的信息块。一个块可以分为块类型、块大小，以及块数据。块类型是 4 字符的 ASCII 码标记，标记中间不能有空格，但末尾可以有。块大小是一个 4 字节(32 位)的值，用于显示块数据的大小。块数据必须占用偶数个字节，必要时可以在结尾补 0。这样，块的每个部分都是文件开头的字排列。块大小不包括块类型和块大小所需要的 8 字节，并且不反映添加的数据。

对于一些块类型，块大小与特定文件无关，是相同的。在块数据是包含信息的固定长度的结构时，就是这种情况。其他情况下，块大小根据特定文件变化。

有两个特殊类型的块分别称为 RIFF 块和 LIST 块。其中，块数据以一个 4 字符 ASCII 形式类型开始，后面是一个或多个子块。LIST 块 RIFF 块类似，只是数据以 4 字符的 ASCII 列表类型开始。RIFF 块用于所有的 RIFF 文件，而 LIST 块只在文件内部用来合并相关子块。

一个 RIFF 文件就是一个 RIFF 块。因此，RIFF 文件以字符串“RIFF”和一个表示文件长度减去 8 字节的 32 位值开始。(实际上，如果需要补充数据则文件可能会长一个字节。)

多媒体 API 包括 16 个前缀是 mmio 的函数，这些函数是专门为 RIFF 文件设计的。DRUMFILE.C 中已经用到其中几个函数来读写 DRUM 数据文件。

要用 mmio 函数打开文件，则第一步是调用 mmioOpen。函数返回一个文件句柄。mmioCreateChunk 函数在文件中创建一个块。它使用 MMCKINFO 定义的块的名称和特征。mmioWrite 函数写块数据。写完块数据以后，调用 mmioAscend。传递给 mmioAscend

的 MMCKINFO 结构必须与前面通过传递给 mmioCreateChunk 来创建块的 MMCKINFO 结构相同。通过从当前文件指针中减去结构的 dwDataOffset 字段来执行 mmioAscend 函数，此文件指针现在位于块数据的结尾，并且此值存储在数据的前面。如果块数据在长度上不是 2 字节的倍数，则 mmioAscend 函数也填补数据。

RIFF 文件由块嵌套级组成。为使 *mmioAscend* 正常工作，必须维护多个 MMCKINFO 结构，每个结构与文件中的一级相联系。DRUM 数据文件共有三级。因此，在 DRUMFILE.C 中的 DrumFileWrite 函数中，我为三个 MMCKINFO 结构定义了一个数组，可以分别标记为 mmckinfo[0]、mmckinfo[1]和 mmckinfo[2]。在第一次 mmioCreateChunk 调用中，mmckinfo[0]结构与 DRUM 形式类型一起用于创建 RIFF 类型的块。其后是第二次 mmioCreateChunk 调用，它用 mmckinfo[1]与 INFO 列表类型一起创建 LIST 类型的块。

第三次 mmioCreateChunk 调用用 mmckinfo[2]创建一个 ISFT 类型的块，此块用于识别创建数据文件的软件。下面的 mmioWrite 调用用于写字符串 szSoftware，调用 mmioAscent 可用 *mmckinfo*[2]来填充此块的块大小字段。这是第一个完整的块。下一个块也在 LIST 块内。程序继续用另一个 mmioCreateChunk 来调用创建 ISCD(creation data:创建数据)块，并再次使用 mmckinfo[2]。在 mmioWrite 调用来写块数据以后，使用 mmckinfo[2]调用 mmioAscend 来填充块大小。那是此块的结尾，也是 LIST 块的结尾。所以，要填充 LIST 块的块大小字段，可再次调用 mmioAscend，这次使用 mmckinfo[1]，它最初用于创建 LIST 块。

要创建“fmt ”和“data”块，mmioCreateChunk 使用 mmckinfo[1]，mmioWrite 调用的后面也使用 mmckinfo[1]的 mmioAscend。在这一点上，除了 RIFF 块本身以外，所有的块大小都填充了。这需要多次使用 *mmckinfo*[0]来调用 *mmioAscend*。虽然有多次调用，但只调用 mmioClose 一次。

看起来好像 mmioAscend 调用改变了当前的文件指针，而且它的确填充了块大小，但在函数返回时，在块数据结束(或可能因补充数据而增加 1 字节)以后，文件指针恢复到以前的位置。从应用程序的观点来看，所有的写文件都是按从头到尾的顺序。

mmioOpen 调用成功之后，除了运行溢出磁盘空间以外不会发生错误。使用变量 wError 从 mmioCreateChunk、mmioWrite、mmioAscend 和 mmioClose 调用累计错误代码，如果磁盘空间不足则每个调用都会失败。如果发生了错误，则 mmioOpen 以 MMIO_DELETE 常量为参数来删除文件，并返回错误信息。

读 RIFF 文件与创建 RIFF 文件类似，只不过是调用 mmioRead 而不是 mmioWrite，调用 mmioDescend 而不是 mmioCreateChunk。“下降”到块，也就是定位块并把文件指针放在块大小之后(或者在 RIFF 或 LIST 块类型的形式类型或者列表类型的后面)。从块“上升”指的是把文件指针移动到块数据的结尾。mmioDescend 和 mmioAscend 函数都不能把文件指针移到文件的前一个位置。

DRUM 以前的版本在 1992 年的《PC Magazine》已发表。那时，Windows 支持两个不

同级的 MIDI 合成器(称为“基本的”和“扩展的”)。那个程序写的文件有格式标识符 1。本章的 DRUM 程序将格式标识符设置为 2。不过,它可以读取并转化早期的格式。这在 DrumFileRead 例程中完成。

第二十三章

领略 Internet

世界范围内的计算机广泛连接，这些计算机为交换信息而执行不同的协议——Internet 近几年又重新定义了个人计算的若干个方面。虽然拨号信息服务和电子邮件系统在 Internet 诞生之前就已经存在，但它们通常局限于字符模式，并且本质上根本没有链接。例如，每个信息服务都需要拨不同的电话号码，用不同的用户 ID 和口令登录。每个电子邮件系统仅允许订阅特定系统的人之间发送和接收邮件。

现在，通常拨一个电话号码就可以连接整个 Internet，而且可以和有电子邮件地址的人进行全球通信。特别是在 World Wide Web 上，超文本、图形和多媒体（包括声音、音乐和视频）的使用已经扩展了在线信息的范围和功能。

覆盖 Windows 中所有与 Internet 相关编程主题的完整指南，还需要另外几本书。实际上，本章主要集中在两个方面，即两个小的 Microsoft Windows 应用程序，用于有效地从 Internet 上获取信息。这两个程序是 Windows 套接字（WinSock：Windows Sockets）API 和 Windows Internet（WinInet）API 的文件传输协议（FTP：File Transfer Protocol）的支持部分。

23.1 Windows 套接字

套接字是由加州大学佰克利分校开发的概念，用于在 UNIX 操作系统上添加网络通信支持。那里开发的 API 现在称为“佰克利（Berkeley）套接字界面”。

23.1.1 套接字和 TCP/IP

套接字一般用于（但不专用于）连接支配 Internet 通信的传输控制协议/网际协议（TCP/IP：Transmission Control Protocol/Internet Protocol）。网际协议（IP：Internet Protocol），作为 TCP/IP 的组成部分之一，包括将数据打包到“数据报（datagram）”，该数据报包含用于标识数据来源和目标的头信息；传输控制协议提供可靠的传输和检查 IP 数据报的方法。

在 TCP/IP 中，通信端点由 IP 地址和端口号定义。IP 地址包括 4 个字节，用于确定 Internet 上的服务器。IP 地址通常按“由点连接的 4 个小于 255 的数字”的格式显示，例如“209.86.105.231”。端口号确定了特定的服务或服务器提供的过程。其中一些端口号已经标准化，以提供众所周知的服务。

当套接字与 TCP/IP 连用时，套接字就是 TCP/IP 的通信端点。因此，套接字指定了 IP 地址和端口号。

23.1.2 网络时间服务

下面给出的示例程序与提供时间协议(Time Protocol)的 Internet 服务器相连接。此程序将获得当前准确的日期和时间，并用此信息设定您的 PC 时钟。

在美国，国家标准和技术协会(National Institute of Standards and Technology)(以前称为国家标准局(National Bureau of Standards))负责维护准确时间，该时间与世界各地的机构相联系。准确时间可用于无线电广播、电话号码、计算机拨号电话号码，以及 Internet，关于这些的所有文档都位于 Web 站点 http://www.bldrdoc.gov/timefreq。(域名“bldrdoc”指的是 Boulder、Colorado、NIST Time 的位置和 Frequency Division。)

我们只对 NIST Network Time Service 感兴趣，其详细的文档位于 http://www.bldrdoc.gov/timefreq/service/nts.htm。此 Web 页列出了 10 个提供 NIST 时间服务的服务器。例如，第一个名称为 time-a.timefreq.bldrdoc.gov，其 IP 地址为 132.163.135.130。

(我曾经编写过一个使用非 Internet NIST 计算机拨号服务的程序，并发表于《PC Magazine》，您也可以在 Ziff-Davis 的 Web 站点 http://www.zdnet.com/pcmag/pctech/content/16/20/ut1620.001.html 中找到。此程序对于想学习如何使用 Windows Telephony API 的人很有帮助。)

在 Internet 上有三个不同的时间服务，每一个都由 Request for Comment(RFC)描述为 Internet 标准。日期协议(Daytime Protocol)(RFC-867)提供了一个 ASCII 字符串用于指出准确的日期和时间。该 ASCII 字符串的准确格式并不标准，但人们可以理解其中的含义。时间协议(RFC-868)提供了一个 32 位的数，用来表示从 1900 年 1 月 1 日至今的秒数。该时间是 UTC(不考虑字母顺序，它表示世界时间坐标(Coordinated Universal Time))，它类似于所谓的格林尼治标准时间(Greenwich Mean Time)或者 GMT——英国格林尼治时间。第三个协议称为“网络时间协议(Network Time Protocol)”(RFC-1305)，该协议很复杂。

对于我们的目的，即包括分析套接字和不断更新 PC 时钟，时间协议已经够用了。RFC-868 只是一个两页的短文档，主要是说用 TCP 获得准确时间的程序应该有如下步骤：

1. 连接到提供此服务的服务器端口 37；

2. 接收 32 位的时间；
3. 关闭连接。

现在我们已经知道了编写一个能访问时间服务的基于套接字的应用程序的每个细节。

23.1.3 NETTIME 程序

Windows 套接字 API，通常也称为 WinSock，与佰克利套接字 API 兼容，因此，可以想象 UNIX 套接字代码可以顺利地传送给 Windows。Windows 下更进一步的支持由对佰克利套接字的扩展提供，其函数的形式是以 WSA（"WinSock API"）为前缀。相关的概述和参考在/Platform SDK/Networking and Distributed Services/Windows Sockets Version 2 中提供。

NETTIME 程序，如程序 23-1 所示，演示了使用 WinSock API 的方法。

程序 23-1 NETTIME 程序

NETTIME.C

```
/* ---------------------------------
    NETTIME.C——Sets System Clock from Internet Services
                (c) Charles Petzold, 1998
   ---------------------------------*/

# include <windows.h>
# include "resource.h"

# define WM_SOCKET_NOTIFY  (WM_USER + 1)
# define ID_TIMER          1

LRESULT  CALLBACK WndProc   (HWND, UINT, WPARAM, LPARAM) ;
BOOL     CALLBACK MainDlg   (HWND, UINT, WPARAM, LPARAM) ;
BOOL     CALLBACK ServerDlg (HWND, UINT, WPARAM, LPARAM) ;

void ChangeSystemTime (HWND hwndEdit, ULONG ulTime) ;
void FormatUpdatedTime (HWND hwndEdit, SYSTEMTIME * pstOld,
                                       SYSTEMTIME * pstNew) ;
void EditPrintf (HWND hwndEdit, TCHAR * szFormat, ...) ;
HINSTANCE  hInst ;
HWND       hwndModeless ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
    static TCHAR  szAppName[] = TEXT ("NetTime") ;
    HWND          hwnd ;
```

```
MSG         msg ;
RECT        rect ;
WNDCLASS    wndclass ;

hInst = hInstance ;

wndclass.style          = 0 ;
wndclass.lpfnWndProc    = WndProc ;
wndclass.cbClsExtra     = 0 ;
wndclass.cbWndExtra     = 0 ;
wndclass.hInstance      = hInstance ;
wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
wndclass.hCursor        = NULL ;
wndclass.hbrBackground  = NULL ;
wndclass.lpszMenuName   = NULL ;
wndclass.lpszClassName  = szAppName ;

if (!RegisterClass (&wndclass))
{
     MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                 szAppName, MB_ICONERROR) ;
     return 0 ;
}

hwnd = CreateWindow (szAppName, TEXT ("Set System Clock from Internet"),
                     WS_OVERLAPPED | WS_CAPTION | WS_SYSMENU |
                          WS_BORDER | WS_MINIMIZEBOX,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     CW_USEDEFAULT, CW_USEDEFAULT,
                     NULL, NULL, hInstance, NULL) ;

    // Create the modeless dialog box to go on top of the window

hwndModeless = CreateDialog (hInstance, szAppName, hwnd, MainDlg) ;

    // Size the main parent window to the size of the dialog box.
    //    Show both windows.

GetWindowRect (hwndModeless, &rect) ;
AdjustWindowRect (&rect, WS_CAPTION | WS_BORDER, FALSE) ;

SetWindowPos (hwnd, NULL, 0, 0, rect.right-rect.left,
              rect.bottom-rect.top, SWP_NOMOVE) ;

ShowWindow (hwndModeless, SW_SHOW) ;
ShowWindow (hwnd, iCmdShow) ;
UpdateWindow (hwnd) ;
```

```
        // Normal message loop when a modeless dialog box is used.

    while (GetMessage (&msg, NULL, 0, 0))
    {
        if (hwndModeless == 0 || !IsDialogMessage (hwndModeless, &msg))
        {
            TranslateMessage (&msg) ;
            DispatchMessage (&msg) ;
        }
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    switch (message)
    {
    case WM_SETFOCUS:
        SetFocus (hwndModeless) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK MainDlg (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static char     szIPAddr[32] = { "132.163.135.130" } ;
    static HWND     hwndButton, hwndEdit ;
    static SOCKET   sock ;
    static struct   sockaddr_in sa ;
    static TCHAR    szOKLabel[32] ;
    int             iError, iSize ;
    unsigned long   ulTime ;
    WORD            wEvent, wError ;
    WSADATA         WSAData ;

    switch (message)
    {
    case WM_INITDIALOG:
        hwndButton = GetDlgItem (hwnd, IDOK) ;
        hwndEdit = GetDlgItem (hwnd, IDC_TEXTOUT) ;
        return TRUE ;

    case WM_COMMAND:
```

```
switch (LOWORD (wParam))
{
case IDC_SERVER:
     DialogBoxParam (hInst, TEXT ("Servers"), hwnd, ServerDlg,
                     (LPARAM) szIPAddr) ;
     return TRUE ;

case IDOK:
          // Call "WSAStartup" and display description text

     if (iError = WSAStartup (MAKEWORD(2,0), &WSAData))
     {
          EditPrintf (hwndEdit, TEXT ("Startup error # %i. \r\n"),
                              iError) ;
          return TRUE ;
     }
     EditPrintf (hwndEdit, TEXT ("Started up %hs \r\n"),
                         WSAData.szDescription);

          // Call "socket"

     sock = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
     if (sock == INVALID_SOCKET)
     {
          EditPrintf (hwndEdit,
                    TEXT ("Socket creation error # %i. \r\n"),
                    WSAGetLastError ()) ;
          WSACleanup () ;
          return TRUE ;
     }
     EditPrintf (hwndEdit, TEXT ("Socket %i created. \r\n"), sock) ;

          // Call "WSAAsyncSelect"

     if (SOCKET_ERROR == WSAAsyncSelect (sock, hwnd, WM_SOCKET_NOTIFY,
                                         FD_CONNECT | FD_READ))
     {
          EditPrintf (hwndEdit,
                    TEXT ("WSAAsyncSelect error # %i. \r\n"),
                    WSAGetLastError ()) ;
          closesocket (sock) ;
          WSACleanup () ;
          return TRUE ;
     }

          // Call "connect" with IP address and time-server port

     sa.sin_family                = AF_INET ;
```

```
        sa.sin_port                 = htons (IPPORT_TIMESERVER) ;
        sa.sin_addr.S_un.S_addr     = inet_addr (szIPAddr) ;

        connect(sock, (SOCKADDR *) &sa, sizeof (sa)) ;

            // "connect" will return SOCKET_ERROR because even if it
            // succeeds, it will require blocking. The following only
            // reports unexpected errors.

        if (WSAEWOULDBLOCK != (iError = WSAGetLastError ()))
        {
             EditPrintf (hwndEdit, TEXT ("Connect error #%i. \r\n"),
                         iError) ;
             closesocket (sock) ;
             WSACleanup () ;
             return TRUE ;
        }
        EditPrintf (hwndEdit, TEXT ("Connecting to %hs..."), szIPAddr) ;

            // The result of the "connect" call will be reported
            // through the WM_SOCKET_NOTIFY message.
            // Set timer and change the button to "Cancel"

        SetTimer (hwnd, ID_TIMER, 1000, NULL) ;
        GetWindowText (hwndButton, szOKLabel, sizeof (szOKLabel) /
                                              sizeof (TCHAR)) ;
        SetWindowText (hwndButton, TEXT ("Cancel")) ;
        SetWindowLong (hwndButton, GWL_ID, IDCANCEL) ;
        return TRUE ;

    case IDCANCEL:
        closesocket (sock) ;
        sock = 0 ;
        WSACleanup () ;
        SetWindowText (hwndButton, szOKLabel) ;
        SetWindowLong (hwndButton, GWL_ID, IDOK) ;

        KillTimer (hwnd, ID_TIMER) ;
        EditPrintf (hwndEdit, TEXT ("\r\nSocket closed.\r\n")) ;
        return TRUE ;
    case IDC_CLOSE:
        if (sock)
            SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;

        DestroyWindow (GetParent (hwnd)) ;
        return TRUE ;
    }
```

```
        return FALSE ;

    case WM_TIMER:
        EditPrintf (hwndEdit, TEXT ("."));
        return TRUE ;

    case WM_SOCKET_NOTIFY:
        wEvent = WSAGETSELECTEVENT (lParam) ;      // ie, LOWORD
        wError = WSAGETSELECTERROR (lParam) ;      // ie, HIWORD

            // Process two events specified in WSAAsyncSelect

        switch (wEvent)
        {
            // This event occurs as a result of the "connect" call

        case FD_CONNECT:
            EditPrintf (hwndEdit, TEXT ("\r\n")) ;

            if (wError)
            {
                EditPrintf (hwndEdit, TEXT ("Connect error # %i."),
                                wError) ;
                SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
                return TRUE ;
            }
            EditPrintf (hwndEdit, TEXT ("Connected to %hs.\r\n"), szIPAddr) ;

                // Try to receive data. The call will generate an error
                // of WSAEWOULDBLOCK and an event of FD_READ

            recv (sock, (char *) &ulTime, 4, MSG_PEEK) ;
            EditPrintf (hwndEdit, TEXT ("Waiting to receive..."));
            return TRUE ;

                // This even occurs when the "recv" call can be made

        case FD_READ:
            KillTimer (hwnd, ID_TIMER) ;
            EditPrintf (hwndEdit, TEXT ("\r\n")) ;

            if (wError)
            {
                EditPrintf (hwndEdit, TEXT ("FD_READ error # %i."),
                                wError) ;
                SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
                return TRUE ;
```

```
                }
                    // Get the time and swap the bytes

                iSize = recv (sock, (char * ) &ulTime, 4, 0) ;
                ulTime = ntohl (ulTime) ;
                EditPrintf (hwndEdit,
                        TEXT ("Received current time of %u seconds")
                        TEXT ("since Jan. 1 1900. \r\n"), ulTime) ;
                    // Change the system time

                ChangeSystemTime (hwndEdit, ulTime) ;
                SendMessage (hwnd, WM_COMMAND, IDCANCEL, 0) ;
                return TRUE ;
            }
            return FALSE ;
        }
        return FALSE ;
}

BOOL CALLBACK ServerDlg (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static char *   szServer ;
    static WORD     wServer = IDC_SERVER1 ;
    char            szLabel [64] ;

    switch (message)
    {
    case WM_INITDIALOG:
        szServer = (char * ) lParam ;
        CheckRadioButton (hwnd, IDC_SERVER1, IDC_SERVER10, wServer) ;
        return TRUE ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDC_SERVER1:
        case IDC_SERVER2:
        case IDC_SERVER3:
        case IDC_SERVER4:
        case IDC_SERVER5:
        case IDC_SERVER6:
        case IDC_SERVER7:
        case IDC_SERVER8:
        case IDC_SERVER9:
        case IDC_SERVER10:
            wServer = LOWORD (wParam) ;
            return TRUE ;
```

```
            case IDOK:
                GetDlgItemTextA (hwnd, wServer, szLabel, sizeof (szLabel)) ;
                strtok (szLabel, "(") ;
                strcpy (szServer, strtok (NULL, ")")) ;
                EndDialog (hwnd, TRUE) ;
                return TRUE ;

            case IDCANCEL:
                EndDialog (hwnd, FALSE) ;
                return TRUE ;
            }
            break ;
        }
        return FALSE ;
}

void ChangeSystemTime (HWND hwndEdit, ULONG ulTime)
{
        FILETIME        ftNew ;
        LARGE_INTEGER   li ;
        SYSTEMTIME      stOld, stNew ;

        GetLocalTime (&stOld) ;

        stNew.wYear         = 1900 ;
        stNew.wMonth        = 1 ;
        stNew.wDay          = 1 ;
        stNew.wHour         = 0 ;
        stNew.wMinute       = 0 ;
        stNew.wSecond       = 0 ;
        stNew.wMilliseconds = 0 ;

        SystemTimeToFileTime (&stNew, &ftNew) ;
        li = * (LARGE_INTEGER * ) &ftNew ;
        li.QuadPart += (LONGLONG) 10000000 * ulTime ;
        ftNew = * (FILETIME * ) &li ;
        FileTimeToSystemTime (&ftNew, &stNew) ;

        if (SetSystemTime (&stNew))
        {
            GetLocalTime (&stNew) ;
            FormatUpdatedTime (hwndEdit, &stOld, &stNew) ;
        }
        else
            EditPrintf (hwndEdit, TEXT ("Could NOT set new date and time.")) ;
}
```

```
void FormatUpdatedTime (HWND hwndEdit, SYSTEMTIME * pstOld, SYSTEMTIME * pstNew)
{
     TCHAR szDateOld [64], szTimeOld [64], szDateNew [64], szTimeNew [64] ;

     GetDateFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSEROVERRIDE | DATE_SHORTDATE,
                    pstOld, NULL, szDateOld, sizeof (szDateOld)) ;

     GetTimeFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSEROVERRIDE |
                         TIME_NOTIMEMARKER | TIME_FORCE24HOURFORMAT,
                    pstOld, NULL, szTimeOld, sizeof (szTimeOld)) ;

     GetDateFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSEROVERRIDE | DATE_SHORTDATE,
                    pstNew, NULL, szDateNew, sizeof (szDateNew)) ;

     GetTimeFormat (LOCALE_USER_DEFAULT, LOCALE_NOUSEROVERRIDE |
                         TIME_NOTIMEMARKER | TIME_FORCE24HOURFORMAT,
                    pstNew, NULL, szTimeNew, sizeof (szTimeNew)) ;

     EditPrintf (hwndEdit,
               TEXT ("System date and time successfully changed ")
               TEXT ("from \r\n\t%s, %s.%03i to \r\n\t%s, %s.%03i."),
               szDateOld, szTimeOld, pstOld -> wMilliseconds,
               szDateNew, szTimeNew, pstNew -> wMilliseconds) ;
}

void EditPrintf (HWND hwndEdit, TCHAR * szFormat, ...)
{
     TCHAR    szBuffer [1024] ;
     va_list pArgList ;

     va_start (pArgList, szFormat) ;
     wvsprintf (szBuffer, szFormat, pArgList) ;
     va_end (pArgList) ;

     SendMessage (hwndEdit, EM_SETSEL, (WPARAM) -1, (LPARAM) -1) ;
     SendMessage (hwndEdit, EM_REPLACESEL, FALSE, (LPARAM) szBuffer) ;
     SendMessage (hwn dEdit, EM_SCROLLCARET, 0, 0) ;
}
```

NETTIME.RC (摘录)

```
// Microsoft Developer Studio generated resource script.

#include "resource.h"
#include "afxres.h"

//////////////////////////////////////////////////////////////////////////////
// Dialog
```

```
SERVERS DIALOG DISCARDABLE 20, 20, 274, 202
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "NIST Time Service Servers"
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON       "OK",IDOK,73,181,50,14
    PUSHBUTTON          "Cancel",IDCANCEL,150,181,50,14
    CONTROL
        "time-a.timefreq.bldrdoc.gov (132.163.135.130) NIST, Boulder, Colorado",
                        IDC_SERVER1,"Button",BS_AUTORADIOBUTTON,9,7,256,16
    CONTROL
        "time-b.timefreq.bldrdoc.gov (132.163.135.131) NIST, Boulder, Colorado",
                        IDC_SERVER2,"Button",BS_AUTORADIOBUTTON,9,24,256,16
    CONTROL
        "time-c.timefreq.bldrdoc.gov (132.163.135.132) Boulder, Colorado",
                        IDC_SERVER3,"Button",BS_AUTORADIOBUTTON,9,41,256,16
    CONTROL
        "utcnist.colorado.edu (128.138.140.44) University of Colorado, Boulder",
                        IDC_SERVER4,"Button",BS_AUTORADIOBUTTON,9,58,256,16
    CONTROL
        "time.nist.gov (192.43.244.18) NCAR, Boulder, Colorado",
                        IDC_SERVER5,"Button",BS_AUTORADIOBUTTON,9,75,256,16
    CONTROL
        "time-a.nist.gov (129.6.16.35) NIST, Gaithersburg, Maryland",
                        IDC_SERVER6,"Button",BS_AUTORADIOBUTTON,9,92,256,16
    CONTROL
        "time-b.nist.gov (129.6.16.36) NIST, Gaithersburg, Maryland",
                        IDC_SERVER7,"Button",BS_AUTORADIOBUTTON,9,109,256,16
    CONTROL
        "time-nw.nist.gov (131.107.1.10) Microsoft, Redmond, Washington",
                        IDC_SERVER8,"Button",BS_AUTORADIOBUTTON,9,126,256,16
    CONTROL
        "utcnist.reston.mci.net (204.70.131.13) MCI, Reston, Virginia",
                        IDC_SERVER9,"Button",BS_AUTORADIOBUTTON,9,143,256,16
    CONTROL
        "nist1.data.com (209.0.72.7) Datum, San Jose, California",
                        IDC_SERVER10,"Button",BS_AUTORADIOBUTTON,9,160,256,16
END

NETTIME DIALOG DISCARDABLE 0, 0, 270, 150
STYLE WS_CHILD
FONT 8, "MS Sans Serif"
BEGIN
    DEFPUSHBUTTON       "Set Correct Time",IDOK,95,129,80,14
    PUSHBUTTON          "Close",IDC_CLOSE,183,129,80,14
    PUSHBUTTON          "Select Server...",IDC_SERVER,7,129,80,14
```

```
    EDITTEXT        IDC_TEXTOUT,7,7,253,110,ES_MULTILINE | ES_AUTOVSCROLL |
                    ES_READONLY | WS_VSCROLL | NOT WS_TABSTOP
END
```

RESOURCE.H（摘录）

```
// Microsoft Developer Studio generated include file.
// Used by NetTime.rc

# define IDC_TEXTOUT            101
# define IDC_SERVER1            1001
# define IDC_SERVER2            1002
# define IDC_SERVER3            1003
# define IDC_SERVER4            1004
# define IDC_SERVER5            1005
# define IDC_SERVER6            1006
# define IDC_SERVER7            1007
# define IDC_SERVER8            1008
# define IDC_SERVER9            1009
# define IDC_SERVER10           1010
# define IDC_SERVER             1011
# define IDC_CLOSE              1012
```

在结构上，NETTIME 程序创建了一个基于 NETTIME.RC 中的 NETTIME 模板的非模态对话框。程序重新定义了窗口的尺寸，以便非模态对话框可以覆盖程序的整个客户区。对话框包括一个只读编辑区（程序用于写文本信息）、一个 Select Server 按钮、一个 Set Correct Time 按钮，以及一个 Close 按钮。Close 按钮用于终止程序。

MainDlg 中的 szIPAddr 变量用于存储服务器地址，默认是字符串“132.163.135.130”。Select Server 按钮激活基于 NETTIME.RC 中的 SERVERS 模板的对话框。szIPAddr 变量作为最后一个参数传递给 DialogBoxParam。Server 对话框列出了 10 个服务器（都是从 NIST Web 站点上逐字复制来的），这些服务器提供了我们感兴趣的服务。当用户单击一个服务器时，ServerDlg 将解析按钮文本，以获得相应的 IP 地址。新地址保存在 szIPAddr 变量中。

当用户按下 Set Correct Time 按钮时，按钮将产生一个 WM_COMMAND 消息，其中 wParam 的低字等于 IDOK。MainDlg 中的 IDOK 处理是最初大部分套接字行为发生的地方。

使用 Windows 套接字 API 时，任何 Windows 程序必须调用的第一个函数是

```
iError = WSAStartup (wVersion, &WSAData) ;
```

NETTIME 将第一个参数设置为 0x0200（表示 2.0 版本）。返回时，WSAData 结构包含了执行 Windows 套接字的信息，而且 NETTIME 将显示 szDescription 字符串，并简要提供了

一些版本信息。

然后，NETTIME 调用 socket 函数：

```
sock = socket (AF_INET, SOCK_STREAM, IPPROTO_TCP) ;
```

第一个参数是一个地址族，表示此处是某种 Internet 地址。第二个参数表示数据以流的形式返回，而不是以数据报的形式返回。（我们需要的数据只有 4 个字节长，而数据报适用于较大的数据块。）最后的参数是一个协议，我们指出是 Internet 协议，即所谓的 TCP。它是 RFC-868 所定义的两个协议之一。socket 函数的返回值存储在 SOCKET 类型的变量中，以便后面的套接字函数的调用。

NETTIME 下面调用的 WSAAsynchSelect 是另一个 Windows 定义的套接字函数。此函数用于避免因 Internet 响应过慢而造成的应用程序挂起。在 WinSock 文档中，有些函数与“组块（blocking）”有关。也就是说，它们不能保证立即把控制返回给程序。WSAAsyncSelect 函数强制通常组块的函数转为非组块，即在函数执行完之前把控制返回给程序。函数的结果以消息的形式报告给应用程序。WSAAsyncSelect 函数让应用程序指定消息和接收消息的窗口的数值。通常，函数的语法如下：

```
WSAAsyncSelect (sock, hwnd, message, iConditions) ;
```

为此任务，NETTIME 使用程序定义的一个消息，该消息称为 WM_SOCKET_NOTIFY。它也用 WSAAsyncSelect 的最后一个参数来指定消息发送的条件，特别是在连接和接收数据时（FD_CONNECT | FD_READ）。

NETTIME 调用的下一个 WinSock 函数是 connect。此函数需要一个指向套接字地址结构的指针，对于不同的协议来说，此套接字地址结构是不同的。NETTIME 使用这一结构的版本是为 TCP/IP 设计的，形如：

```
struct sockaddr_in
{
    short           sin_family;
    u_short         sin_port;
    struct in_addr  sin_addr;
    char            sin_zero[8];
};
```

其中 in_addr 是用于指定 Internet 地址的联合，它可以用 4 个字节，或者 2 个无符号短整数，或者 1 个无符号长整数来表示。

NETTIME 将 sin_family 字段设置为 AF_INET，用于表示地址族；将 sin_port 设置为端口号，这里是时间协议的端口号，RFC-868 显示为 37。但不要像我最初时那样，将此字段设为 37。当大多数数字通过 Internet 时，结构的这个端口号字段必须是“大终点”，即最重要的字节必须是第一个。Intel 微处理器是小终点。幸运的是，htons（“host-to-net-

work short")函数使字节翻转,因此 NETTIME 将 sockaddr_in 结构的 sin_port 字段设置为:

```
htons (IPPORT_TIMESERVER)
```

WINSOCK2.H 中将常数定义为 37。NETTIME 用 inet_addr 函数将存储在 szIPAddr 字符串中的服务器地址转化为无符号长整数,该整数用于设置结构的 sin_addr 字段。

如果应用程序在 Windows 98 下调用 connect,而且当前 Windows 没有连接到 Internet,那么将显示"拨号连接"对话框。这就是所谓的"自动拨号"。在 Windows NT 4.0 中不能实现"自动拨号",因此如果在 NT 环境下运行,那么在运行 NETTIME 之前,必须先连接 Internet。

connect 函数通常要受阻,这是因为连接之前它需要占用一些时间。然而,由于 NETTIME 调用了 WSAAsyncSelect,所以 connect 不会等待连接,事实上,它会立即返回 SOCKET_ERROR 的值。这并不是出现了错误,所有运行的函数都表明连接没有成功,甚至 NETTIME 也不会检查此返回值。相反,它调用 WSAGetLastError。如果 WSAGetLastError 返回 WSAEWOULDBLOCK(即函数通常要受阻,但这里并没有受阻),那么一切都很好。NETTIME 将 Set Correct Time 按钮改成 Cancel,并设置了一个 1 秒的计时器。WM_TIMER 处理只在程序窗口中显示周期,以向用户指出程序仍在运行,没有使系统崩溃。

连接最终完成时,MainDlg 由 WM_SOCKET_NOTIFY 消息——NETTIME 在 WSAAsyncSelect 函数中指定的程序定义的消息所通知。lParam 的低字等于 FD_CONNECT,高字表示错误。这时的错误可能是程序不能连接到指定的服务器。NETTIME 还列出了其他 9 个服务器,供您选择,可以试试其他的服务器!

如果一切顺利,那么 NETTIME 将调用 recv("receive(接收)")函数来读取数据:

```
recv (sock, (char *) &ulTime, 4, MSG_PEEK) ;
```

这意味着,用 4 个字节来存储 ulTime 变量。最后一个参数表示只是"窥视"此数据,并不将其从输入队列中删除。像 connect 函数一样,recv 返回一个错误代码,以表示函数通常受阻,但这时没有受阻。理论上讲(当然这不大可能),函数至少能返回数据的一部分,然后通过再次调用以获得其余的 32 个字节值。那就是调用 recv 函数时带有 MSG_PEEK 选项的原因。

与 connect 函数类似,recv 函数也产生 WM_SOCKET_NOTIFY 消息,这时带有 FD_READ 的事件代码。NETTIME 通过再次调用 recv 来对此响应,这时最后的参数是 0,用于从队列中删除数据。我将简要讨论一下程序处理接收到的 ulTime 的方法。注意,NETTIME 通过向自己发送带有 wParam 的 WM_COMMAND 消息来结束处理,此时 wParam 取值为 IDCANCEL。对话框过程通过调用 closesocket 和 WSACleanup 来响应。

请回忆一下 NETTIME 接收的 32 位的 ulTime 值是从 1990 年 1 月 1 日 0:00UTC 开始

的秒数。但最重要字节是第一个字节，因此该值必须通过 ntohl("network-to-host long") 函数处理来记录字节，以便 Intel 微处理器能够处理，然后，NETTIME 调用 ChangeSystemTime 函数。

ChangeSystemTime 以当前的本地时间开始——调整到用户所在时区和白天存储时间的当前系统时间，然后将 SYSTEMTIME 结构设置为 1900 年 1 月 1 日午夜(0 时)，并将这个 SYSTEMTIME 结构传递给 SystemTimeToFileTime，将此结构转化为 FILETIME 结构。FILETIME 实际上只是由两个 32 位的 DWORD 一起组成 64 位的整数，用来表示从 1601 年 1 月 1 日至今间隔为 100×10^{-9} 秒的间隔数。

ChangeSystemTime 函数将 FILETIME 结构转化为 LARGE_INTEGER，它是一个联合，允许 64 位的值可以按两个 32 位的值引用，或者按一个基于__int64 数据类型的 64 位整数引用。(__int64 数据类型是 Microsoft 编译程序对 ANSI C 标准的扩展。)因此，此值是 1601 年 1 月 1 日 ~ 1900 年 1 月 1 日之间间隔为 100×10^{-9} 秒的间隔数。这里，添加了 1900 年 1 月 1 日至当前间隔为 100×10^{-9} 秒的间隔数——ulTime 的 1 000 万倍。

然后通过调用 FileTimeToSystemTime 将作为结果而生成的 FILETIME 值转换回 SYSTEMTIME 结构。因为间协议返回当前的 UTC 时间，所以 NETTIME 通过调用 SetSystemTime 来设置时间，SetSystemTime 也基于 UTC。出于显示的目的，程序调用 GetLocalTime 来获得更新时间。最初的本地时间和新的本地时间一起传递给 FormatUpdatedTime，这个函数用 GetTimeFormat 函数和 GetDateFormat 函数将时间转化为 ASCII 字符串。

如果程序在 Windows NT 下运行，并且用户没有设置时间的权限，那么 SetSystemTime 函数可能失败。如果 SetSystemTime 失败，则 NETTIME 将发布一则"未设置新时间"的消息以指出有问题发生了。

23.2 WinInet 和 FTP

WinInet("Windows Internet")API 是一个高级函数集，这些函数将用三个普通的 Internet 协议来帮助程序员，这三个协议是：用于 World Wide Web 的超文本传输协议(HTTP：Hypertext Transfer Protocol)、文件传输协议(FTP：File Transfer Protocol)和另一个被称为 Gopher 的文件协议。WinInet 函数的语法与常用的 Windows 文件函数的语法类似，这使得使用这些协议就像使用本地磁盘上的文件一样容易。WinInet API 的文档位于/Platform SDK/Internet, Intranet, Extranet Services/Internet Tools and Technologies/WinInet API。

下面的示例程序将演示如何使用 WinInet API 的 FTP 部分。许多有 Web 站点的公司也都有"匿名 FTP"站点，这样用户可以在不输入用户名和口令的情况下下载文件。例如，如果在 Internet Explorer 的地址栏输入 ftp://ftp.microsoft.com，那么就可以浏览目录并下载文件。如果进入 ftp://ftp.cpetzold.com/cpetzold.com/ProgWin/UpdDemo，那么将在我的匿名 FTP 站点上发现用于连接示例程序的文件列表，下面将详细讨论此示例

程序。

虽然现今 FTP 对大多数的 Web 用户来说并不那么友好,但它仍然相当有用。例如,应用程序能利用 FTP 从匿名 FTP 站点上获取数据,这些 FTP 站点几乎完全位于屏幕后面,很少受到用户的干涉。这就是我们将讨论的 UPDDEMO("update demonstration:更新示例")程序的指导思想。

23.2.1 FTP API 概况

使用 WinInet 的程序必须在所有调用 WinInet 函数的源文件中包括头文件 WININET.H,程序还必须链接 WININET.LIB。在 Microsoft Visual C++ 中,您可以在 Project Settings 对话框的 Link 选项卡中指定。运行时,程序将和 WININET.DLL 动态链接库链接。

在下面的论述中,我不会详细讨论函数的语法,因为某些函数有很多选项,这导致它相当复杂。要掌握 WinInet,您可以将 UPDDEMO 源代码作为食谱来对待。这时最重要的是了解有关的各个步骤以及 FTP 函数的范围。

要使用 Windows Internet API,首先要调用 InternetOpen。在对这个函数进行一次调用之后您能使用 WinInet 支持的任意一种协议。InternetOpen 给您一个 Internet 会话句柄,并保存到 HINTERNET 类型的变量中。用完 WinInet API 以后,应该通过调用 InternetCloseHandle 来关闭句柄。

为了使用 FTP,接下来您就要调用 InternetConnect。此函数需要由 InternetOpen 创建 Internet 会话句柄,并且把一个句柄返回给 FTP。对于以前缀 Ftp 开始的所有函数,您都可以将这一句柄作为函数的第一个参数使用。InternetConnect 函数的参数指出您想要使用 FTP,而且提供诸如 ftp.cpetzold.com 之类的服务器名称。此函数还需要用户名和口令。如果访问匿名 FTP 站点,这些参数可以设置为 NULL。如果应用程序调用 InternetConnect 时 PC 并没有连接到 Internet,Windows 98 将显示"拨号连接"对话框。当使用 FTP 的应用程序结束时,调用 InternetCloseHandle 来关闭句柄。

这时可以开始调用有 Ftp 前缀的函数。您将发现,这些函数与标准的 Windows 文件 I/O 函数很相似。为了避免与其他协议重复,一些以 Internet 为前缀的函数也可以处理 FTP。

下面 4 个函数用于处理目录:

```
fSuccess = FtpCreateDirectory (hFtpSession, szDirectory) ;
fSuccess = FtpRemoveDirectory (hFtpSession, szDirectory) ;
fSuccess = FtpSetCurrentDirectory (hFtpSession, szDirectory) ;
fSuccess = FtpGetCurrentDirectory (hFtpSession, szDirectory,
                                   &dwCharacterCount) ;
```

注意,这些函数很像我们所熟悉的 Windows 提供用于处理本地文件系统的 CreateDirectory、RemoveDirectory、SetCurrentDirectory 和 GetCurrentDirectory 函数。

当然,访问匿名 FTP 站点的应用程序不能创建或删除目录,而且,程序也不能假定 FTP 目录具有和 Windows 文件系统相同的目录结构类型。特别是用相对路径名设置目录的程序,不能假定关于新的目录全名称的一切。如果程序需要知道结果目录的整个名称,那么调用了 SetCurrentDirectory 之后必须再调用 GetCurrentDirectory。GetCurrentDirectory 的字符串参数至少包含 MAX_PATH 字符,并且最后一个参数应指向包含该值的变量。

下面两个函数允许您删除或重新命名文件(但不是在匿名 FTP 站点上):

```
fSuccess = FtpDeleteFile (hFtpSession, szFileName) ;
fSuccess = FtpRenameFile (hFtpSession, szOldName, szNewName) ;
```

通过先调用 FtpFindFirstFile,可以查找一个文件(或与含有通配符的模板相匹配的多个文件)。此函数很像 FindFirstFile 函数,甚至使用了相同的 WIN32_FIND_DATA 结构。该文件为文件列举返回了一个句柄。您可以将此句柄传递给 InternetFindNextFile 函数以获得附加文件名。最后通过调用 InternetCloseHandle 来关闭句柄。

要打开文件,可以调用 FtpFileOpen。这个函数返回一个文件句柄,此句柄可以用于 InternetReadFile、InternetReadFileEx、InternetWrite 和 InternetSetFilePointer 调用。最后可以通过调用最常用的 InternetCloseHandle 函数来关闭句柄。

最后,下面两个高级函数特别有用:FtpGetFile 调用将文件从 FTP 服务器复制到本地存储器,它合并了 FtpFileOpen、FileCreate、InternetReadFile、WriteFile、InternetCloseHandle 和 CloseHandle 调用。FtpGetFile 的一个参数是一个标记,如果本地已经存在同名文件,那么该标记将导致函数调用失败;FtpPutFile 与此函数类似,用于将文件从本地存储器复制到 FTP 服务器。

23.2.2　更新演示程序

如程序 23-2 所示 UPDDEMO 的,演示了用 WinInet FTP 函数在第二个线程执行期间从匿名 FTP 站点下载文件的方法。

程序 23-2　UPDDEMO 程序

UPDDEMO.C

```
/*-------------------------
   UPDDEMO.C——Demonstrates Anonymous FTP Access
              (c) Charles Petzold, 1998
   -------------------------*/

#include <windows.h>
#include <wininet.h>
```

```
# include < process.h >
# include "resource.h"

    // User-defined messages used in WndProc

# define WM_USER_CHECKFILES (WM_USER + 1)
# define WM_USER_GETFILES   (WM_USER + 2)

    // Information for FTP download

# define FTPSERVER TEXT ("ftp.cpetzold.com")
# define DIRECTORY TEXT ("cpetzold.com/ProgWin/UpdDemo")
# define TEMPLATE TEXT ("UD??????.TXT")

    // Structures used for storing filenames and contents

typedef struct
{
     TCHAR * szFilename ;
     char  * szContents ;
}
FILEINFO ;

typedef struct
{
     int       iNum ;
     FILEINFO  info[1] ;
}
FILELIST ;

    // Structure used for second thread

typedef struct
{
     BOOL bContinue ;
     HWND hwnd ;
}
PARAMS ;

    // Declarations of all functions in program

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;
BOOL    CALLBACK DlgProc (HWND, UINT, WPARAM, LPARAM) ;
VOID             FtpThread (PVOID) ;
VOID             ButtonSwitch (HWND, HWND, TCHAR * ) ;
FILELIST *       GetFileList (VOID) ;
int              Compare (const FILEINFO * , const FILEINFO * ) ;
```

```
    // A couple globals

HINSTANCE hInst ;
TCHAR     szAppName[ ] = TEXT ("UpdDemo") ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                    PSTR szCmdLine, int iCmdShow)
{
     HWND        hwnd ;
     MSG         msg ;
     WNDCLASS    wndclass ;

     hInst = hInstance ;

     wndclass.style          = 0 ;
     wndclass.lpfnWndProc    = WndProc ;
     wndclass.cbClsExtra     = 0 ;
     wndclass.cbWndExtra     = 0 ;
     wndclass.hInstance      = hInstance ;
     wndclass.hIcon          = LoadIcon (NULL, IDI_APPLICATION) ;
     wndclass.hCursor        = NULL ;
     wndclass.hbrBackground  = GetStockObject (WHITE_BRUSH) ;
     wndclass.lpszMenuName   = NULL ;
     wndclass.lpszClassName  = szAppName ;

     if (!RegisterClass (&wndclass))
     {
          MessageBox (NULL, TEXT ("This program requires Windows NT!"),
                      szAppName, MB_ICONERROR) ;
          return 0 ;
     }

     hwnd = CreateWindow (szAppName, TEXT ("Update Demo with Anonymous FTP"),
                          WS_OVERLAPPEDWINDOW | WS_VSCROLL,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          CW_USEDEFAULT, CW_USEDEFAULT,
                          NULL, NULL, hInstance, NULL) ;

     ShowWindow (hwnd, iCmdShow) ;
     UpdateWindow (hwnd) ;

          // After window is displayed, check if the latest file exists

     SendMessage (hwnd, WM_USER_CHECKFILES, 0, 0) ;

     while (GetMessage (&msg, NULL, 0, 0))
     {
          TranslateMessage (&msg) ;
```

```
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
    static FILELIST * plist ;
    static int        cxClient, cyClient, cxChar, cyChar ;
    HDC               hdc ;
    int               i ;
    PAINTSTRUCT       ps ;
    SCROLLINFO        si ;
    SYSTEMTIME        st ;
    TCHAR             szFilename [MAX_PATH] ;

    switch (message)
    {
    case WM_CREATE:
        cxChar = LOWORD (GetDialogBaseUnits ()) ;
        cyChar = HIWORD (GetDialogBaseUnits ()) ;
        return 0 ;

    case WM_SIZE:
        cxClient = LOWORD (lParam) ;
        cyClient = HIWORD (lParam) ;

        si.cbSize = sizeof (SCROLLINFO) ;
        si.fMask = SIF_RANGE | SIF_PAGE ;
        si.nMin  = 0 ;
        si.nMax  = plist ? plist -> iNum - 1 : 0 ;
        si.nPage = cyClient / cyChar ;

        SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
        return 0 ;

    case WM_VSCROLL:
        si.cbSize = sizeof (SCROLLINFO) ;
        si.fMask  = SIF_POS | SIF_RANGE | SIF_PAGE ;
        GetScrollInfo (hwnd, SB_VERT, &si) ;

        switch (LOWORD (wParam))
        {
        case SB_LINEDOWN:       si.nPos += 1 ;          break ;
        case SB_LINEUP:         si.nPos -= 1 ;          break ;
        case SB_PAGEDOWN:       si.nPos += si.nPage ;   break ;
        case SB_PAGEUP:         si.nPos -= si.nPage ;   break ;
```

```
            case SB_THUMBPOSITION:    si.nPos = HIWORD (wParam) ;    break ;
            default:                  return 0 ;
            }
            si.fMask = SIF_POS ;
            SetScrollInfo (hwnd, SB_VERT, &si, TRUE) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_USER_CHECKFILES:
                // Get the system date & form filename from year and month

            GetSystemTime (&st) ;
            wsprintf (szFilename, TEXT ("UD%04i%02i.TXT"), st.wYear, st.wMonth) ;

                // Check if the file exists; if so, read all the files

            if (GetFileAttributes (szFilename) != (DWORD) -1)
            {
                SendMessage (hwnd, WM_USER_GETFILES, 0, 0) ;
                return 0 ;
            }
                // Otherwise, get files from Internet.
                // But first check so we don't try to copy files to a CD-ROM!

            if (GetDriveType (NULL) == DRIVE_CDROM)
            {
                MessageBox (hwnd, TEXT ("Cannot run this program from CD-ROM!"),
                            szAppName, MB_OK | MB_ICONEXCLAMATION) ;
                return 0 ;
            }
                // Ask user if an Internet connection is desired

            if (IDYES == MessageBox (hwnd,
                                     TEXT ("Update information from Internet?"),
                                     szAppName, MB_YESNO | MB_ICONQUESTION))

                // Invoke dialog box

            DialogBox (hInst, szAppName, hwnd, DlgProc) ;

                // Update display

            SendMessage (hwnd, WM_USER_GETFILES, 0, 0) ;
            return 0 ;

        case WM_USER_GETFILES:
            SetCursor (LoadCursor (NULL, IDC_WAIT)) ;
```

```
            ShowCursor (TRUE) ;

                // Read in all the disk files

            plist = GetFileList () ;

            ShowCursor (FALSE) ;
            SetCursor (LoadCursor (NULL, IDC_ARROW)) ;

                // Simulate a WM_SIZE message to alter scroll bar & repaint

            SendMessage (hwnd, WM_SIZE, 0, MAKELONG (cxClient, cyClient)) ;
            InvalidateRect (hwnd, NULL, TRUE) ;
            return 0 ;

        case WM_PAINT:
            hdc = BeginPaint (hwnd, &ps) ;
            SetTextAlign (hdc, TA_UPDATECP) ;

            si.cbSize = sizeof (SCROLLINFO) ;
            si.fMask = SIF_POS ;
            GetScrollInfo (hwnd, SB_VERT, &si) ;

            if (plist)
            {
                for (i = 0 ; i < plist -> iNum ; i++)
                {
                    MoveToEx (hdc, cxChar, (i - si.nPos) * cyChar, NULL) ;
                    TextOut   (hdc, 0, 0, plist -> info[i].szFilename,
                                  lstrlen (plist -> info[i].szFilename)) ;
                    TextOut   (hdc, 0, 0, TEXT (": "), 2) ;
                    TextOutA  (hdc, 0, 0, plist -> info[i].szContents,
                                  strlen (plist -> info[i].szContents)) ;
                }
            }
            EndPaint (hwnd, &ps) ;
            return 0 ;

        case WM_DESTROY:
            PostQuitMessage (0) ;
            return 0 ;
        }
        return DefWindowProc (hwnd, message, wParam, lParam) ;
}

BOOL CALLBACK DlgProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM lParam)
{
```

```
    static PARAMS params ;

    switch (message)
    {
    case WM_INITDIALOG:
        params.bContinue = TRUE ;
        params.hwnd = hwnd ;

        _beginthread (FtpThread, 0, &params) ;
        return TRUE ;

    case WM_COMMAND:
        switch (LOWORD (wParam))
        {
        case IDCANCEL:              // button for user to abort download
            params.bContinue = FALSE ;
            return TRUE ;
        case IDOK:                  // button to make dialog box go away
            EndDialog (hwnd, 0) ;
            return TRUE ;
        }
    }
    return FALSE ;
}

/* ------------------------------------------
    FtpThread: Reads files from FTP server and copies them to local disk
   ------------------------------------------ */

void FtpThread (PVOID parg)
{
    BOOL            bSuccess ;
    HINTERNET       hIntSession, hFtpSession, hFind ;
    HWND            hwndStatus, hwndButton ;
    PARAMS          * pparams ;
    TCHAR           szBuffer [64] ;
    WIN32_FIND_DATA finddata ;

    pparams = parg ;
    hwndStatus = GetDlgItem (pparams -> hwnd, IDC_STATUS) ;
    hwndButton = GetDlgItem (pparams -> hwnd, IDCANCEL) ;

        // Open an internet session

    hIntSession = InternetOpen (szAppName, INTERNET_OPEN_TYPE_PRECONFIG,
                                NULL, NULL, INTERNET_FLAG_ASYNC) ;
    if (hIntSession == NULL)
    {
```

```
        wsprintf (szBuffer, TEXT ("InternetOpen error %i"), GetLastError ()) ;
        ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
        _endthread () ;
    }

    SetWindowText (hwndStatus, TEXT ("Internet session opened...")) ;

        // Check if user has pressed Cancel

    if (!pparams -> bContinue)
    {
        InternetCloseHandle (hIntSession) ;
        ButtonSwitch (hwndStatus, hwndButton, NULL) ;
        _endthread () ;
    }

        // Open an FTP session.

    hFtpSession = InternetConnect (hIntSession, FTPSERVER,
                                   INTERNET_DEFAULT_FTP_PORT,
                                   NULL, NULL, INTERNET_SERVICE_FTP, 0, 0) ;
    if (hFtpSession == NULL)
    {
        InternetCloseHandle (hIntSession) ;
        wsprintf (szBuffer, TEXT ("InternetConnect error %i"),
                            GetLastError ()) ;
        ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
        _endthread () ;
    }

    SetWindowText (hwndStatus, TEXT ("FTP Session opened...")) ;

        // Check if user has pressed Cancel

    if (!pparams -> bContinue)
    {
        InternetCloseHandle (hFtpSession) ;
        InternetCloseHandle (hIntSession) ;
        ButtonSwitch (hwndStatus, hwndButton, NULL) ;
        _endthread () ;
    }

        // Set the directory

    bSuccess = FtpSetCurrentDirectory (hFtpSession, DIRECTORY) ;

    if (! bSuccess)
    {
```

```
        InternetCloseHandle (hFtpSession) ;
        InternetCloseHandle (hIntSession) ;
        wsprintf (szBuffer, TEXT ("Cannot set directory to %s"),
                  DIRECTORY) ;
        ButtonSwitch (hwndStatus, hwndButton, szBuffer) ;
        _endthread () ;
    }

    SetWindowText (hwndStatus, TEXT ("Directory found...")) ;

        // Check if user has pressed Cancel
    if (!pparams -> bContinue)
    {
        InternetCloseHandle (hFtpSession) ;
        InternetCloseHandle (hIntSession) ;
        ButtonSwitch (hwndStatus, hwndButton, NULL) ;
        _endthread () ;
    }
        // Get the first file fitting the template

    hFind = FtpFindFirstFile (hFtpSession, TEMPLATE,
                  &finddata, 0, 0) ;

    if (hFind == NULL)
    {
        InternetCloseHandle (hFtpSession) ;
        InternetCloseHandle (hIntSession) ;
        ButtonSwitch (hwndStatus, hwndButton, TEXT ("Cannot find files")) ;
        _endthread () ;
    }

    do
    {
            // Check if user has pressed Cancel

        if (!pparams -> bContinue)
        {
            InternetCloseHandle (hFind) ;
            InternetCloseHandle (hFtpSession) ;
            InternetCloseHandle (hIntSession) ;
            ButtonSwitch (hwndStatus, hwndButton, NULL) ;
            _endthread () ;
        }
            // Copy file from internet to local hard disk, but fail
            // if the file already exists locally

        wsprintf (szBuffer, TEXT ("Reading file %s..."), finddata.cFileName) ;
        SetWindowText (hwndStatus, szBuffer) ;
```

```
            FtpGetFile (hFtpSession,
                        finddata.cFileName, finddata.cFileName, TRUE,
                        FILE_ATTRIBUTE_NORMAL, FTP_TRANSFER_TYPE_BINARY, 0) ;
        }
        while (InternetFindNextFile (hFind, &finddata)) ;
        InternetCloseHandle (hFind) ;
        InternetCloseHandle (hFtpSession) ;
        InternetCloseHandle (hIntSession) ;

        ButtonSwitch (hwndStatus, hwndButton, TEXT ("Internet Download Complete"));
}

/*----------------------------------------
    ButtonSwitch: Displays final status message and changes Cancel to OK
   ----------------------------------------*/

VOID ButtonSwitch (HWND hwndStatus, HWND hwndButton, TCHAR * szText)
{
    if (szText)
        SetWindowText (hwndStatus, szText) ;
    else
        SetWindowText (hwndStatus, TEXT ("Internet Session Cancelled")) ;

    SetWindowText (hwndButton, TEXT ("OK")) ;
    SetWindowLong (hwndButton, GWL_ID, IDOK) ;
}

/*----------------------------------------
    GetFileList: Reads files from disk and saves their names and contents
   ----------------------------------------*/

FILELIST * GetFileList (void)
{
    DWORD           dwRead ;
    FILELIST      * plist ;
    HANDLE          hFile, hFind ;
    int             iSize, iNum ;
    WIN32_FIND_DATA finddata ;

    hFind = FindFirstFile (TEMPLATE, &finddata) ;

    if (hFind == INVALID_HANDLE_VALUE)
        return NULL ;

    plist = NULL ;
    iNum = 0 ;
```

```
    do
    {
            // Open the file and get the size

        hFile = CreateFile (finddata.cFileName, GENERIC_READ, FILE_SHARE_READ,
                            NULL, OPEN_EXISTING, 0, NULL) ;
        if (hFile == INVALID_HANDLE_VALUE)
            continue ;

        iSize = GetFileSize (hFile, NULL) ;

        if (iSize == (DWORD) -1)
        {
            CloseHandle (hFile) ;
            continue ;
        }
            // Realloc the FILELIST structure for a new entry

        plist = realloc (plist, sizeof (FILELIST) + iNum * sizeof (FILEINFO));

            // Allocate space and save the filename

        plist -> info[iNum].szFilename = malloc (lstrlen (finddata.cFileName) +
                                                 sizeof (TCHAR)) ;
        lstrcpy (plist -> info[iNum].szFilename, finddata.cFileName) ;

            // Allocate space and save the contents

        plist -> info[iNum].szContents = malloc (iSize + 1) ;
        ReadFile (hFile, plist -> info[iNum].szContents, iSize, &dwRead, NULL);
        plist -> info[iNum].szContents[iSize] = 0 ;

        CloseHandle (hFile) ;
        iNum ++ ;
    }
    while (FindNextFile (hFind, &finddata)) ;

    FindClose (hFind) ;

        // Sort the files by filename

    qsort (plist -> info, iNum, sizeof (FILEINFO), Compare) ;

    plist -> iNum = iNum ;

    return plist ;
}
```

```
/* - - - - - - - - - - - - -
    Compare function for qsort
   - - - - - - - - - - - - - */

int Compare (const FILEINFO * pinfo1, const FILEINFO * pinfo2)
{
    return lstrcmp (pinfo2 -> szFilename, pinfo1 -> szFilename) ;
}
```

UPDDEMO.RC (摘录)

```
//Microsoft Developer Studio generated resource script.

# include "resource.h"
# include "afxres.h"

////////////////////////////////////////////////////////////////////////////
// Dialog

UPDDEMO DIALOG DISCARDABLE 20, 20, 186, 95
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Internet Download"
FONT 8, "MS Sans Serif"
BEGIN
    PUSHBUTTON          "Cancel",IDCANCEL,69,74,50,14
    CTEXT               "",IDC_STATUS,7,29,172,21
END
```

RESOURCE.H (摘录)

```
// Microsoft Developer Studio generated include file.
// Used by UpdDemo.rc

# define IDC_STATUS             40001
```

UPDDEMO 使用的文件名称是 UDyyyymm.TXT,其中 yyyy 是 4 位阿拉伯数字的年数(当然适用于 2000),mm 是 2 位阿拉伯数字的月数。这里假定程序从每月更新的文件中获益。这些文件可能是整个月刊,而由于性能上的考虑,程序将其下载到本地存储器。

因此,WinMain 在调用 ShowWindow 和 UpdateWindow 来显示 UPDDEMO 主窗口以后,向 WndProc 发送程序定义的 WM_USER_CHECKFILES 消息。WndProc 通过获得当前的年、月并检查该年月 UDyyyymm.TXT 文件所在的默认目录来处理此消息。这种文件的存在意义在于 UPDDEMO 是不断更新的。(当然,事实并非如此。一些过时的文件将丢失。更完整的程序可以进行更广泛的检测。)在这种情况下,UPDDEMO 向自己发送一个 WM_USER_GETFILES 消息,它通过调用 GetFileList 函数来处理该消息。这是 UPDDEMO.C 中稍长一点的函数,但它并不是特别有用,它所做的全部工作就是将所有的

UDyyyymm.TXT 文件读到动态分配的 FILELIST 类型的结构，该结构在程序顶部定义。然后程序在其客户区显示这些文件的内容。

如果 UPDDEMO 没有最新的文件，那么它必须访问 Internet 以进行更新。程序首先询问用户这样做是否 OK。如果是，程序将显示一个简单的对话框，其中只有一个 Cancel 按钮和一个 ID 为 IDC_STATUS 的静态文本区。下载时，此静态文本区向用户提供状态报告，并且允许用户取消过于缓慢的会话。此对话过程的名称是 DlgProc。

DlgProc 很短，它建立了一个包括自身窗口句柄的 PARAMS 类型的结构以及一个名称为 bContinue 的 BOOL 变量，然后调用_beginthread 来执行第二个线程。

FtpThread 函数通过使用下面的调用来完成实际的转换：InternetOpen、InternetConnect、FtpSetCurrentDirectory、FtpFindFirstFile、InternetFindNextFile、FtpGetFile 和 InternetCloseHandle(三次)。由于有很多代码，所以如果此线程函数不需要检查错误、让用户了解下一步的操作情况，以及允许用户随意取消整个显示，那么它将简洁得多。FtpThread 函数通过用 hwndStatus 句柄调用 SetWindowText 来让用户知道进展情况，这里指的是对话框中间的静态文本区。

线程可以按下面的三种方式之一来终止：

第一种方式，FtpThread 可能遇到从 WinInet 函数返回的错误。如果是这样，它将清除并编排错误字符串的格式，然后将此字符串(连同对话框文本区句柄和 Cancel 按钮的句柄一起)传递给 ButtonSwitch。ButtonSwitch 是一个小函数，它显示了文本字符串，并将 Cancel 按钮转换成 OK 按钮——不只是按钮上的文本字符串的转换，还包括控制 ID 的转换。这样就允许用户按下 OK 按钮来结束对话框。

第二种方式，FtpThread 可以在没有任何错误的情况下完成任务，其处理方法和遇到错误时的方法一样，只不过对话框中显示的字符串为 Internet Download Complete。

第三种方式，用户可以在过程中选择取消下载。这时，DlgProc 将 PARAMS 结构的 bContinue 字段设置为 FALSE。FtpThread 频繁地检查该值，如果 bContinue 等于 FALSE，那么函数将清除并以 NULL 文本参数调用 ButtonSwitch，此参数表示字符串 Internet Session Cancelled 已被显示。同样，用户必须按下 OK 按钮来关闭对话框。

虽然 UPDDEMO 只能显示每个文件的一行，但我可以用这个程序来告诉您任何可能的更新，以及关于本书的其他信息，您也可以在 Web 站点上发现更详细的信息。因此，UPDDEMO 成为我向您传送信息的手段，并且可以作为本书最后一页的继续。